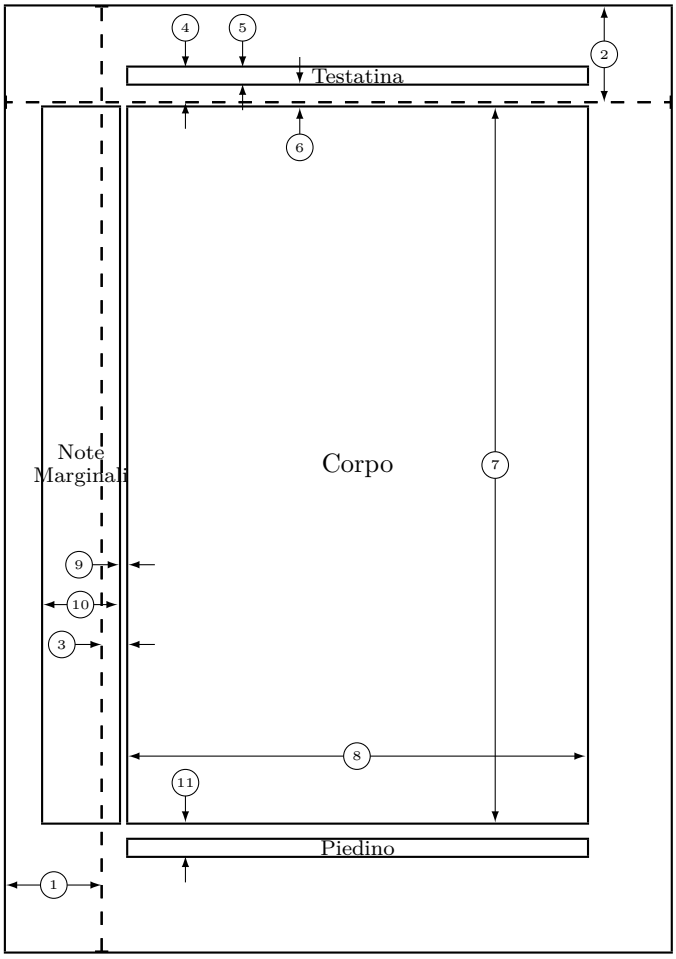
Gruppo Utilizzatori Italiani di TEX

Introduzione all'arte della composizione tipografica con LATEX

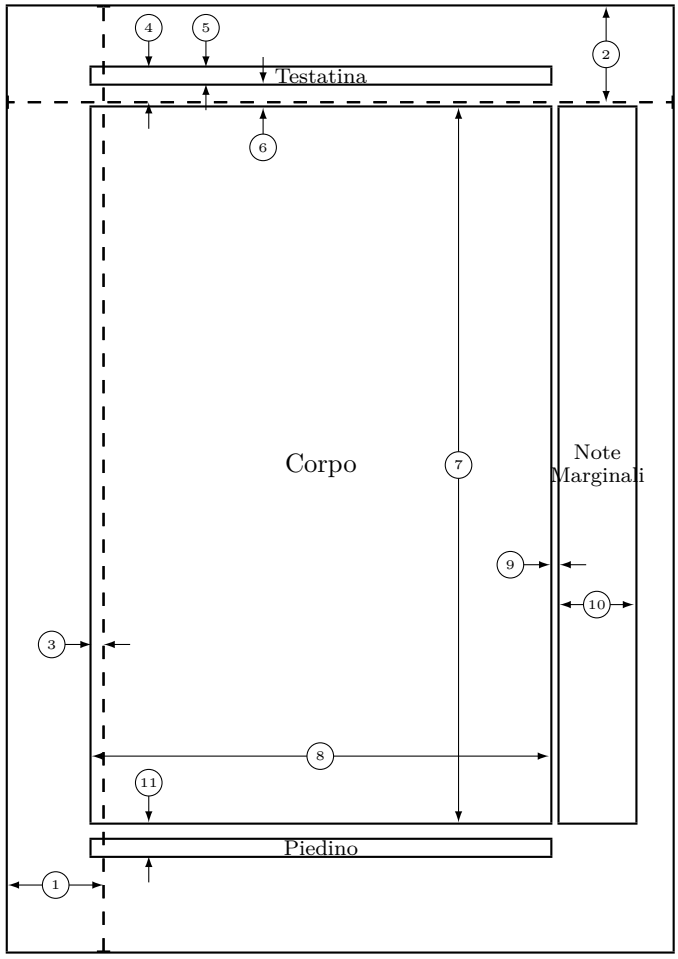
Tomo III

16 maggio 2015

Introduzione all'arte della composizione tipografica con LaTeX



1 un pollice + \hoffset
2 un pollice + \voffset
3 \evensidemargin = 20pt
4 \topmargin = -26pt
5 \headheight = 12pt
6 \headsep = 18pt
7 \textheight = 538pt
8 \textwidth = 345pt
9 \marginparsep = 7pt
10 \marginparwidth = 57pt
\marginparpush = 5pt (non mostrato)
11 \footskip = 25pt
\hoffset = 0pt
\voffset = 0pt
\paperwidth = 500pt
\paperheight = 711pt



1	un pollice + \hoffset	2	un pollice + \voffset
3	\oddsidemargin = -9pt	4	\topmargin = -26pt
5	\headheight = 12pt	6	\headsep = 18pt
7	\textheight = 538pt	8	\textwidth = 345pt
9	\marginparsep = 7pt	10	\marginparwidth = 57pt
11	\footskip = 25pt		\marginparpush = 5pt (non mostrato)
	\hoffset = 0pt		\voffset = 0pt
	\paperwidth = 500pt		\paperheight = 711pt

Introduzione all'arte della composizione tipografica con LaTeX

GuIt

Versione B5-0.99i — 2015/05/16
Pdf minor version = 5

Quest'opera è soggetta alla Creative Commons Public License versione 2.5 o posteriore. L'enunciato integrale della Licenza in versione 2.5 è reperibile all'indirizzo internet `http://creativecommons.org/licenses/by-nc-nd/3.0/deed.it`.

- Si è liberi di riprodurre, distribuire, comunicare al pubblico, esporre in pubblico, rappresentare, eseguire e recitare quest'opera alle seguenti condizioni:

  Attribuzione Bisogna attribuire la paternità dell'opera nei modi indicati dall'autore o da colui al quale è stata data quest'opera in licenza; in questo caso si tratta del Gruppo degli Utilizzatori Italiani di TEX, GUIT.

  Non commerciale Non si può usare quest'opera per fini commerciali.

  Non opere derivate Non si può alterare o trasformare quest'opera, né usarla per crearne un'altra.

- Ogni volta che si usa o si distribuisce quest'opera, lo si deve fare secondo i termini di questa licenza, che va comunicata con chiarezza.

- In ogni caso si possono concordare con il titolare dei diritti d'autore (il GUIT, in questo caso) usi di quest'opera in deroga da questa licenza.

I nomi commerciali, i loghi, i trademark appartengono ai rispettivi proprietari.

La foto 2.2 della pagina 11 e le foto delle pagine 13, 229, e 432 — 434 sono di Claudio Beccari.
L'immagine 2.1 della pagina 11 è stata ricavata da un indirizzo Internet non più reperibile; l'autore sembra essere Hermann Zapf.

Coordinatore: Claudio Beccari

Hanno collaborato a questo testo in modo diretto o indiretto: Alessandro Andretta, Luciano Battaia, Francesco Biccari, Paolo Biffis, Riccardo Campana, David Carlisle, Antonio Cervone, Gustavo Cevolani, Agostino De Marco, Roberto Giacomelli, Tommaso Gordini, Enrico Gregorio, Orlando Iovino, Maurizio Himmelmann, Jerónimo Leal, Antonio Macrì, Giuseppe Molteni, Federico Morchio, Lorenzo Pantieri, Gianluca Pignalberi, Ottavio Rizzo, Salvatore Schirone, Luigi Scarso, Andrea Tonelli, Ivan Valbusa, Emanuele Zannarini.

Un grazie particolare a Enrico Gregorio.

Presentazione

Questo testo è rivolto a coloro che amano la bella composizione tipografica e trovano che i testi composti con LaTeX siano molto professionali. Questo implica che il lettore tipo abbia già un certa dimestichezza con LaTeX e abbia letto, almeno in parte, o il testo introduttivo [46] o, meglio ancora, l'ottimo manuale di Lorenzo Pantieri e Tommaso Gordini *L'arte di scrivere con LaTeX*, [49].

Alcune parti di questo testo possono risultare un po' ostiche a chi è completamente digiuno di programmazione, ma si è fatto ogni sforzo per rendere comprensibile queste parti anche a chi ha poca dimestichezza con il calcolatore. Certo c'è un compromesso fra chiarezza e concisione, ma si spera che il lettore vorrà ugualmente comprendere l'impegno del gruppo di persone che ha dato vita a questo testo.

Questo testo sulla composizione tipografica mediante LaTeX è stato infatti predisposto da diversi membri del Gruppo degli Utilizzatori Italiani di TeX, GuIt, il cui obiettivo è proprio quello di far conoscere il sistema di composizione sviluppato dal matematico Donald E. Knuth più di trenta anni fa. Non si tratta di un programma di interesse archeologico, perché esso è vivo e vegeto, ha dato origine a un buon numero di discendenti e alcune sue parti sono usate all'interno di altri programmi di elaborazione di testi, senza che gli utenti di questi programmi lo sappiano.

Il sistema TeX è stato uno dei primi esempi di software libero; questa sua qualità ne ha permesso il contributo creativo e/o critico di una moltitudine di utenti, come succede sempre con il software libero, per cui si è arricchito nel tempo di una moltitudine di estensioni che gli permettono di comporre praticamente qualsiasi cosa, tranne, forse, certi tipi di pieghevoli pubblicitari.

Si tratta di un programma di composizione tipografica, non di un impaginatore; quest'ultimo tipo di programmi consente di agire sul materiale da impaginare come se fosse una figura da modificare o da adattare, anche se svolge in parte le funzioni di compositore.

LaTeX, un elemento del sistema TeX, consente di comporre tipograficamente dei testi contenenti testo corrente, sia in prosa sia in poesia, scritti in qualunque alfabeto, per esempio latino, greco, cirillico, dall'andamento diretto (da sinistra a destra), oppure ebraico, arabo, dall'andamento inverso (da destra a sinistra), oppure cinese, giapponese, coreano, dall'andamento anche verticale.

Può gestire font di ogni genere, sia quelli a matrici di punti, sia i font PostScript, sia i font TrueType, OpenType, eccetera. Tali font possono essere codificati in varie maniere, ma alcuni 'figli' di TEX gestiscono anche i font codificati secondo la norma UNICODE.

La caratteristica che più differenzia il sistema TEX dagli altri elaboratori di composizione è il fatto che per comporre un documento con questo sistema bisogna agire in tempi diversi per introdurre il testo e per comporlo; in questo non è molto diverso da certi procedimenti professionali di impaginazione, dove il testo da comporre viene introdotto in un file di solo testo che poi viene in un secondo tempo fatto fluire dentro il programma di impaginazione, assemblandolo insieme alle figure e all'altro materiale non testuale per generare il documento 'finito', pronto da inviare alla fotoincisione e alla stampa.

Invece questo modo di comporre è molto diverso da quello dei word processor, dove il compositore vede direttamente sullo schermo del suo elaboratore il testo già composto, così da poter esaminare immediatamente il frutto del suo lavoro; l'analisi di questa differenza nel modo di procedere verrà svolta nel primo capitolo.

Il secondo capitolo darà al lettore alcune nozioni di tipografia, se non altro per abituarlo alle alcune parole che ricorrono spesso nella descrizione delle varie operazioni compositive. Chi avesse già queste nozioni può saltare la lettura di questo capitolo, ma se, nonostante tutto, decidesse di leggerlo, potrebbe constatare che alcune nozioni e/o alcuni vocaboli in questa guida sono usati per indicare cose leggermente diverse da quelle che conosceva.

Il terzo capitolo è dedicato all'ortografia tipografica; oltre alla definizione, che ovviamente non contrasta con l'ortografia della lingua, vengono impostate anche le buone regole per l'uso della sillabazione, delle forme dei caratteri, dell'uso della punteggiatura, dell'elisione e del troncamento, sottolineando che la grammatica consente numerose varianti, ma la buona tipografia predilige l'omogeneità e l'adesione a consuetudini ormai acquisite; la buona tipografia aiuta il lettore a leggere senza affaticarsi; usare convenzioni diverse dalle 'solite' affatica il lettore che deve abituarsi a comprendere quel che legge, ma che è scritto in modo per lui insolito.

Il quarto capitolo esaminerà le procedure da seguire per procurarsi il software del sistema TEX e per installare i programmi e i file accessori; non si vuole mettere in ombra la centralità dei programmi di composizione del sistema TEX, ma bisogna attrarre l'attenzione sui programmi accessori, per esempio gli *shell editor* la cui scelta può risultare determinante nella facilità con cui si può ottenente il meglio dal sistema TEX.

I capitoli cinque, sette ed otto richiamano i primi rudimenti della composizione asincrona mediante LATEX (parte del sistema TEX). Si parlerà essenzialmente di composizione testuale, di liste e di tabelle. Queste ultime già possono dare qualche pensiero anche al lettore con una modesta esperienza alle spalle.

LATEX si distingue da alcuni suoi fratelli, figli del sistema TEX, per consentire al compositore, oltre che allo scrittore, di concentrarsi sul messaggio da trasmettere al lettore, invece che sulla forma da dargli, cioè sulla sua estetica. A seconda del documento da comporre lo stile di composizione può essere molto diverso; anche

la semplice impaginazione può assumere aspetti grafici diversi; LaTeX consente di scegliere in modo globale il tipo di documento e di apportare piccole modifiche stilistiche durante la composizione, senza che il compositore debba preoccuparsi della nerezza dei caratteri, oppure della distanza del numero indicativo di un paragrafo dall'inizio del suo titolino. A tutte queste cose pensa la scelta iniziale del tipo di documento e con le opzioni specificate. Il capitolo sei si occupa appunto di queste cose.

I capitoli nove e dieci si dedicano alla creazione, manipolazione e inclusione di figure.

Passando ad argomenti più specializzati, i capitoli undici e dodici si occupano della preparazione della bibliografia e della composizione di indici analitici e glossari.

I capitoli tredici e quattordici si dedicano invece alla composizione della matematica; se il lettore ha familiarità con manuali di tipografia, avrà notato che l'argomento della composizione della matematica è praticamente assente da quei manuali, perché si tratta di un tipo di composizione abbastanza specializzato e che interviene raramente nei libri pubblicati. Ma un documento non è necessariamente un libro di narrativa; può essere un rapporto tecnico, una tesi di laurea o di dottorato, un manuale o un prontuario tecnico, un articolo scientifico, insomma, uno scritto dove la matematica compare spesso in forma avanzata.

Alla composizione di testi letterari e di filologia è dedicato il capitolo quindici; questi testi verosimilmente non contengono una sola formula matematica, ma richiedono un tipo di composizione spesso bidimensionale, che, come la matematica, richiede di comprendere a fondo i meccanismi compositivi bidimensionali. Alle necessità compositive della filologia si provvede mediante l'uso di moduli esterni, chiamati *pacchetti di estensione*, che fanno parte integrante del sistema TeX.

Il capitolo sedici si riferisce alla preparazione delle presentazioni, cioè di quei documenti che verranno verosimilmente usati per essere proiettati mentre se ne espone il contenuto a voce durante una conferenza, una lezione, o simili.

Il capitolo diciassette tratta dell'arte della composizione tipografica con LaTeX; questo è proprio il capitolo che giustifica il titolo di questo manuale; vi si parlerà delle tecniche di cui LaTeX o, più ancora, pdfLaTeX dispongono per eseguire la composizione tipografica come e meglio della totalità dei word processor e meglio di molti programmi di impaginazione; si tratta della microgiustificazione, una tecnica a cui ricorrevano fin dall'inizio i prototipografi, a cominciare da Gutenberg, ma di cui si è un po' persa l'abitudine nel corso dei secoli successivi, sia per i costi, sia per i vincoli imposti dai caratteri metallici; per fortuna la composizione elettronica è svincolata da queste limitazioni, ma ovviamente bisogna che il programma di composizione sia stato programmato per la microgiustificazione.

Il capitolo diciotto si riferisce ai caratteri da stampa e ne descrive le particolarità, i comandi per gestirli, i modi per caricare altri caratteri; fornisce qualche nozione di tipografia, ma dedica ampio spazio ai font del sistema TeX, il quale, è bene ribadirlo, non è limitato ai font disponibili in prima installazione, che

sono già molto numerosi, ma può usare virtualmente qualsiasi font disponibile sia gratuito, sia commerciale.

Il capitolo diciannove è dedicato all'arte di definire nuovi comandi compositivi per agevolare il compito del compositore.

Il capitolo venti tratta invece del progetto grafico delle pagine e delle strutture che compongono il documento; normalmente queste sono decisioni e scelte operate dal grafico editoriale; con LaTeX spesso siamo noi stessi i grafici editoriali, i compositori, talvolta anche gli editori dei nostri documenti; è bene avere anche una buona formazione in questo senso, almeno fino al livello di conoscere i problemi, senza avere la presunzione di saperli risolvere come un professionista. Conoscendo i problemi, però, può nascere il desiderio di conoscere più a fondo l'arte tipografica, che ci porterà non solo ad apprezzare maggiormente i prodotti tipografici ben riusciti, ma anche ad arricchire la nostra cultura in un settore spesso trascurato o considerato erroneamente troppo tecnico per meritare il nome di 'arte'. Ricordiamo invece che il nome del sistema TeX è ottenuto dalle prime tre lettere maiuscole della parola greca $\tau\acute{\epsilon}\chi\nu\eta$[1], che, come ricorda Knuth stesso, vuol dire *arte*.

Questo testo non è un manuale; per LaTeX ce ne sono di liberi e di commerciali assai validi; a questo argomento è dedicata il capitolo ventuno. Nello stesso tempo, dicendo che questo non è un manuale si vorrebbe sottolineare che le indicazioni qui esposte servono per affrontare la composizione con LaTeX con un approccio che guarda di più alla sostanza, alla composizione professionale, che non all'aspetto grafico del singolo dettaglio man mano che questo si presenta, come succede spesso, invece, con i vari word processor.

Per esempio in questo testo si parla di 'scatole' proprio di sfuggita, mentre in ogni manuale questo argomento richiede almeno una sezione espressamente dedicata loro. Così si parla poco o nulla di contatori o di lunghezze rigide o elastiche; sono argomenti importanti, ma che servono maggiormente per scrivere i programmi, le macroistruzioni, contenute nei file che specificano lo stile compositivo o nei file che raccolgono le macro personali. Durante la composizione non bisognerebbe mai perdersi in questi dettagli.

Piuttosto la programmazione in linguaggio TeX può diventare essenziale per l'utente che deve comporre testi dallo stile insolito o che abbia bisogno di strutture compositive particolari.

Qui, però, tranne una esposizione sommaria nel capitolo 19, non si parlerà di queste cose, ma si rinvia direttamente il lettore a manuali che trattano questo aspetto con maggiore dettaglio, (capitolo 21). Si veda piuttosto il capitolo 29, di tipo più manualistico, dove sono raccolte, commentate, modificate le informazioni che Leslie Lamport ha scritto nel suo manuale [35].

Il capitolo ventidue è dedicato ad un aspetto piuttosto moderno e fortemente legato alla tipografia elettronica, vale a dire si occupa del problema dell'archiviabilità dei documenti elettronici; questa caratteristica richiede strumenti ed

[1] $\tau\acute{\epsilon}\chi\nu\eta$ in Italia viene pronunciata 'tècne'; in Grecia 'téchni'; in ogni caso quella che sembra una X nel nome di TeX e di LaTeX va pronunciata come una 'k' o come una forte espirazione, non come una 'x'.

accorgimenti particolari che si possono raggiungere con l'uso del sistema TEX, ma che richiedono comunque una speciale attenzione.

Il capitolo ventitre riguarda invece un aspetto produttivo che coinvolge la tipografia elettronica; si tratta del lavoro di assemblaggio dei file da comporre con il sistema TEX, segnatamente con LATEX e pdfLATEX, quando i file sorgente non hanno il mark-up di LATEX. Si può eseguire l'elaborazione manuale, ma è meglio disporre di adeguati strumenti per la conversione automatica, almeno per ottenere un primo approccio al mark-up di LATEX, ma con la trasformazione delle codifiche dei vari caratteri in modo da non alterare il messaggio del testo da comporre e, specialmente, per non introdurre errori durante il processo di conversione. L'editoria moderna deve poter usare questi strumenti con cognizione di causa, sia per il lavoro commerciale sia per quello che gradatamente si va estendendo, cioè la documentazione libera, la produzione di testi liberamente scaricabili dalla rete, ma composti con ogni cura al massimo livello qualitativo.

Il capitolo 24 è dedicata alla simbologia e alla nomenclatura delle grandezze fisiche, nonché ai simboli codificati dalle norme ISO per l'uso nella matematica usata dai fisici e dai tecnologi; vi sono anche non pochi riferimenti al Sistema Internazionale delle Unità di Misura e alla loro 'ortografia'.

Il capitolo 25 spiega invece in ogni dettaglio l'algoritmo che viene usato dai programmi del sistema TEX per dividere le parole in fin di riga; questo è un procedimento che in italiano funziona impeccabilmente nella totalità dei casi, ma è bene conoscerne i dettagli per intervenire in quei pochi casi in cui la giustificazione non viene eseguita perfettamente oppure quando si scrive in una lingua straniera dalle regole più rigide di quelle che valgono per l'italiano; oppure per ricorrere ad una sillabazione italiana diversa da quella prescritta dalla norme UNI, ma consentita in casi particolari.

Il capitolo 26 vorrebbe riempire un vuoto di informazione che per usare LATEX va assolutamente colmato. Si tratta della distinzione fra la codifica di entrata e di quella di uscita; fra la codifica che l'utente, tramite i suoi programmi, usa per scrivere un testo usando la tastiera, e quella che il programma di composizione usa in relazione ai font impiegati. Bisogna conoscere queste due codifiche, distinguerle e usarle con cognizione di causa.

Il capitolo 27, molto tecnico, cerca di spiegare i dettagli di come ogni motore di composizione del sistema TEX costruisca i capoversi e costruisca le pagine; si tratta di un meccanismo abbastanza complesso, ma la sua comprensione aiuta a capire perché la composizione talvolta sia diversa da come ce la aspetteremmo. Normalmente ha ragione il motore di composizione a fare quello che fa, ma talvolta può essere necessaria una piccola dose di aiuto consapevole.

Il capitolo 29, lungi dal rappresentare una semplice traduzione della corrispondente appendice del manuale [35] di Leslie Lamport, il creatore di LATEX, pur mantenendone l'impostazione, rappresenta, si spera, una fonte comoda e pratica per rivedere in modo ragionato ma non troppo prolisso la sintassi dei vari comandi del mark-up di LATEX. In questa parte, per la verità, si sono saltate rispetto al manuale di Lamport, intere parti che oggi sono considerate superate; invece alcune parti sono state estese considerevolmente.

Il lettore non si scoraggi se vengono usate molte parole inglesi; in un certo senso la cosa è voluta, perché i comandi del *mark-up* di LaTeX sono quasi tutti formati da parole inglesi o da loro abbreviazioni; in ogni punto in cui questi comandi vengono descritti si è sempre cercato di associare le parole inglesi alla spiegazione italiana; tuttavia non si è rifuggiti da alcune similitudini insolite, come 'flottante' (che esiste in italiano con un significato simile) corrispondente a *floating*, ritenuto preferibile rispetto all'italianissima parola 'mobile' che esprime lo stesso concetto, ma che forse è espresso con una parola slegata dai messaggi che LaTeX espone sullo schermo durante la sua esecuzione. Così si è frequentemente usata la locuzione *di default*, piuttosto che usare espressioni italiane come 'predefinito', 'da usarsi in mancanza d'altro', 'se non si è specificato diversamente', e simili locuzioni quasi equivalenti l'una all'altra.

Inoltre, sebbene non si sia sicuri di essere stati sempre consistenti, è stata usata la congiunzione oppositiva 'o' per indicare il concetto esclusivo di 'o questo o quello ma non entrambi', e la congiunzione 'oppure' per indicare il concetto inclusivo di 'o questo o quello o entrambi'. La lingua parlata e quella scritte sono flessibili e talvolta non è necessario essere così precisi, ma si cerchi di tenere presente questo particolare uso delle congiunzioni oppositive.

Di fianco a molte figure vi è una piccola legenda, spesso in verticale, con il nome dell'autore e il nome del programma, quasi sempre del sistema TeX, che l'autore ha usato. Questa indicazione non è messa vicino alle fotografie oppure alle immagini tratte da Internet; la paternità di queste immagini è indicata, ove possibile, nel retrofrontespizio. Per le altre figure, oltre a indicare l'autore del disegno mostrato, si è ritenuto utile indicare anche il programma usato per realizzare il disegno; quasi tutti i disegni sono stati realizzati usando comandi e mark-up forniti da pacchetti di estensione del programma di composizione; questi pacchetti consentono di eseguire egregiamente molti tipi di disegni, senza bisogno di ricorrere a programmi esterni; hanno anche il vantaggio di essere stilisticamente compatibili con la tipografia eseguita con LaTeX, cosa che spesso non è possibile ottenere con i normali word processor o con altri programmi di disegno assistito dal calcolatore.

Il materiale di questa guida è stato riordinato rispetto alla edizioni precedenti; ovviamente è anche stato aggiornato e l'aggiornamento continuo cerca di non restare indietro rispetto all'evoluzione continua del sistema TeX; non è possibile garantire che questo testo sia sempre aggiornato all'ultima distribuzione del sistema TeX, ma si fa il possibile per farlo.

Questa guida può essere composta in formato A4 (210 mm per 297 mm) o in formato B5 (176 mm per 250 mm); quest'ultimo formato può venire stampato e rifilato in modo che il volume brossurato a colla o legato possa trovare facilmente spazio nei normali scaffali; può venire anche stampato in tre tomi distinti in formato B5, dove però le pagine iniziali (dal frontespizio fino agli indici) e le pagine finali (dalla bibliografia in poi) siano comuni a tutti i tomi; i riferimenti degli indici generale e analitici saranno quindi sempre corretti anche in riferimento a tomi diversi.

Indice

Presentazione IX

1 Composizione sincrona e asincrona 1
1.1 Introduzione . . . 1
1.2 Il mark-up di LaTeX . . . 2
1.3 Scribus: un impaginatore libero con licenza GPL . . . 3
1.4 Conclusione . . . 7

2 Nozioni elementari di tipografia 9
2.1 Tipografia e dattilografia . . . 9
2.2 Unità di misura tipografiche . . . 10
2.3 Misure tipografiche . . . 13
2.4 Le particolarità dei caratteri . . . 15
2.5 I contrografismi . . . 17
2.6 Le parti di alcuni documenti a stampa . . . 18
2.7 Osservazioni finali . . . 20

3 Ortografia tipografica 21
3.1 Ortografia testuale . . . 21
3.1.1 Ortografie alternative . . . 22
3.1.2 'd' eufonica e 'i' prostetica, apostrofi . . . 23
3.2 Accenti . . . 24
3.3 Sillabazione . . . 26
3.4 Punteggiatura . . . 27
3.5 Abbreviazioni . . . 33
3.6 Appellativi e maiuscole . . . 35
3.7 Uso dei font . . . 36
3.8 Le note . . . 37
3.9 Conclusioni . . . 38

4 Installare il sistema TeX 41
4.1 Installazione su macchine Windows da XP in poi . . . 42
4.1.1 Installare MiKTeX . . . 42

4.1.2 Installare TEX Live 43
4.2 Installazione su Linux 44
4.3 Installazione su Macintosh con Mac OS X 45
4.4 Gli alberi di cartelle del sistema TEX 48
4.4.1 Gli alberi di cartelle di TEX Live 50
4.4.2 Gli alberi di cartelle di MiKTEX 51
4.4.3 L'aggiornamento dei database dei nomi dei file 53
4.5 I programmi accessori 54
4.6 Le tastiere 56
4.6.1 Le tastiere delle macchine Windows 57
4.6.2 Tastiere Linux 60
4.6.3 Le tastiere sulle macchine Macintosh 60
4.6.4 Le tastiere virtuali 61
4.7 Gli shell editor 62
4.7.1 Shell editor multipiattaforma 65
4.7.2 Shell editor per le macchine Windows 68
4.7.3 Shell editor per le macchine Linux 70
4.7.4 Shell editor per le macchine Macintosh 73
4.8 Editor quasi WYSIWYG 77
4.9 LATEX e pdfLATEX 78
4.10 Altri programmi del sistema TEX 81
4.10.1 Plain TEX 82
4.10.2 I programmi estesi 82
4.10.3 Il mark-up ConTEXt 84
4.10.4 I programmi Omega e Lambda, Aleph e Lamed 85
4.10.5 Il programma XETEX 85
4.10.6 Il programma LuaTEX 86
4.11 Il sistema TEX 88

5 LATEX: prime nozioni 93
5.1 Introduzione 93
5.2 L'inizio del file sorgente 93
5.3 Il documento 95
5.4 La fine del documento 96
5.5 Un semplice esercizio 96
5.6 I caratteri speciali 98
5.7 Testo strutturato 100
5.8 Organizzazione dei file sorgente 101
5.9 Gestione degli errori 108

6 LATEX: i vari tipi di documenti e stili di composizione 111
6.1 Introduzione 111
6.2 Classi standard 111
6.3 La creazione di nuove classi 113
6.4 Alcune classi non standard 114

6.4.1 Le classi Komascript 114
6.4.2 La classe *memoir* 115
6.4.3 La classe *ncc* 116
6.4.4 Le classi per le tesi di laurea 117
6.4.5 L'estensione `layaureo` 119
6.5 I pacchetti di estensione 120
6.5.1 Come invocare i file di estensione 121
6.5.2 I vari pacchetti e gli archivi internazionali 122
6.6 Come scrivere nuovi pacchetti 123
6.7 Non modificare i pacchetti esistenti 126

7 LaTeX: testi speciali 131
7.1 Che cosa sono i testi in display 131
7.2 Le citazioni 131
7.2.1 Le citazioni brevi 132
7.2.2 Le citazioni lunghe 133
7.2.3 I versi 134
7.2.4 Brani in linguaggi speciali 135
7.3 Gli elenchi 136
7.3.1 Le elencazioni in linea 136
7.3.2 Le enumerazioni 136
7.3.3 Le elencazioni semplici 137
7.3.4 Alcune osservazioni relative alle elencazioni 138
7.4 Le descrizioni 139
7.5 Le liste bibliografiche 139
7.6 I riferimenti incrociati 141
7.7 Altri testi in display 142
7.8 Le note 145
7.8.1 Le note in calce 145
7.8.2 Le note marginali 147
7.9 Un esempio specifico di testi speciali 148

8 LaTeX: tabelle 153
8.1 Introduzione 153
8.2 Come far flottare una tabella 154
8.3 Le didascalie 154
8.4 Come comporre la tabella vera e propria 155
8.4.1 I descrittori delle colonne 156
8.4.2 Il raggruppamento delle celle 157
8.4.3 I separatori verticali 159
8.4.4 Come rendere le tabelle un poco più aperte 160
8.5 Le tabelle di larghezza specificata 162
8.6 Problemi compositivi delle tabelle 165
8.6.1 Tabelle troppo larghe 165
8.7 Tabelle troppo lunghe 167

8.8 Pacchetti di estensione per le tabelle 168

9 LaTeX: figure **175**
9.1 Le figure e le immagini 175
9.2 L'ambiente `figure` 175
9.2.1 Controllo dei grandi oggetti flottanti 176
9.2.2 Modifica degli ambienti flottanti 176
9.3 L'ambiente `picture` 177
9.4 Il pacchetto **pgf** 185
9.5 Vantaggi dei programmi nativi del sistema TeX 188
9.6 METAPOST 189
9.7 Usi insoliti dell'ambiente *picture* 191
9.8 Linee guida per la grafica 193
9.8.1 Preliminari 194
9.8.2 Il tempo necessario per la creazione della grafica 195
9.8.3 Piano di lavoro per creare un grafico 196
9.8.4 Collegamento fra testo e grafico 196
9.8.5 Coerenza fra testo e grafica 198
9.8.6 Legende nei grafici 199
9.8.7 Diagrammi di vario genere 199
9.8.8 Attenzione e distrazione 205
9.8.9 Commenti 206

10 LaTeX: l'importazione di figure esterne **209**
10.1 Introduzione 209
10.2 I formati grafici 209
10.2.1 I formati vettoriali 209
10.2.2 I formati diversi da quelli vettoriali 210
10.3 I formati accettabili 212
10.3.1 I formati accettabili da latex 214
10.3.2 I formati accettabili da pdflatex 216
10.4 Conversione dei formati 218
10.5 Scontornare le immagini 220
10.6 L'importazione delle immagini 221
10.6.1 Organizzare le immagini 222
10.6.2 Includere le immagini 225

11 La bibliografia **231**
11.1 Introduzione 231
11.2 I programmi di estrazione bibliografica 232
11.2.1 Come specificare lo stile bibliografico 232
11.2.2 Come comporre la bibliografia 232
11.2.3 Chiavi e citazioni 233
11.3 I database bibliografici 235
11.4 Quanti database bibliografici? 237

11.5 Il pacchetto *biblatex* 239

12 LaTeX: indici e glossari **241**
12.1 Introduzione 241
12.2 L'indice analitico 241
12.2.1 Il programma makeindex 242
12.2.2 La composizione effettiva dell'indice analitico 243
12.3 Il glossario 244
12.4 Modifica dell'indice analitico 245
12.5 Indicizzazione sincrona 245
12.5.1 Indicizzazione sincrona dell'indice analitico 247
12.5.2 Indicizzazione sincrona del glossario 248
12.6 Composizione automatica dell'indice analitico 248

13 LaTeX: la matematica semplice **251**
13.1 Introduzione 251
13.2 I modi matematici 251
13.3 Alcune annotazioni sulle lettere greche 256
13.4 Alcune osservazioni sugli operatori funzionali 257
13.5 Alcune osservazioni sui grandi operatori 260
13.6 I grandi delimitatori 261
13.7 Accenti e segni diacritici matematici 262
13.8 Gli ambienti matematici 263
13.9 Le unità di misura 265

14 LaTeX: la matematica avanzata **269**
14.1 I simboli di *amsmath* 269
14.2 Le estensioni dei font matematici 272
14.3 I sistemi di equazioni 272
14.4 Gli ambienti di composizione di *amsmath* 273
14.4.1 L'ambiente *equation* 274
14.4.2 L'ambiente *aligned* 274
14.4.3 L'ambiente *split* 275
14.4.4 L'ambiente *multline* 276
14.4.5 L'ambiente *gather* 277
14.4.6 L'ambiente *align* 277
14.4.7 L'ambiente *flalign* 278
14.4.8 L'ambiente *alignat* 279
14.4.9 L'ambiente *subequations* 279
14.4.10 Gli ambienti spezzati 280
14.5 Altri comandi e ambienti 281
14.5.1 Definizione di operatori funzionali 281
14.5.2 Le frazioni in generale e le frazioni continue 282
14.5.2.1 Le frazioni e gli altri costrutti simili 282
14.5.2.2 Le frazioni continue 282

14.5.3 Il testo intercalato alle equazioni 282
14.5.4 Le frecce estensibili 284
14.5.5 Gli indici incolonnati 284
14.5.6 Gli integrali multipli 284
14.5.7 L'operatore differenziale 285
14.5.8 I simboli corsivi matematici in nero 285
14.5.9 Le espressioni matematiche riquadrate 286
14.6 Le matrici e i determinanti 287
14.7 I diagrammi commutativi 290
14.8 La punteggiatura in matematica 291
14.9 Conclusioni 295

15 LaTeX: composizione di testi letterari e filologici 297
15.1 La composizione di testi letterari 297
15.2 Scrivere in greco 301
15.3 Font adatti ai testi classici 304
15.4 La composizione del tedesco classico 305
15.5 La composizione di testi in lingue classiche 307
15.5.1 Scrivere in latino 307
15.5.2 Scrivere in greco 308
15.6 La composizione di testi filologici 311
15.7 Un esempio di composizione con XeLaTeX 314
15.7.1 Esempio di composizione in greco 314
15.7.2 XeLaTeX, i font OpenType e i font Type 1 316
15.8 Conclusione 319

16 LaTeX: le presentazioni 321
16.1 Introduzione 321
16.2 Le classi per le presentazioni 321
16.3 Altre classi per le presentazioni 323
16.4 La classe `beamer` 323
16.5 La documentazione 324
16.6 Una breve presentazione 325
16.7 Creare un nuovo stile 331
16.8 Osservazioni 332

17 LaTeX: la microgiustificazione 333
17.1 Introduzione 333
17.2 La composizione dei capoversi 334
17.3 Metodi per migliorare la giustificazione 336
17.4 La microgiustificazione 337
17.5 Come funziona la microtipografia 339

18 LaTeX: i caratteri da stampa **343**
18.1 Introduzione . . . 343
18.2 Terminologia relativa ai caratteri . . . 343
18.3 I comandi per la scelta dei font . . . 352
18.3.1 La scelta del corpo e dell'avanzamento di riga . . . 352
18.3.2 La scelta delle altre caratteristiche . . . 357
18.4 Altri font diversi da quelli di default . . . 359
18.5 I font per la matematica . . . 360
18.6 Il Text Companion Font . . . 366
18.7 Gli alfabeti diversi da quello latino . . . 367
18.7.1 Caratteri cirillici . . . 367
18.7.2 Caratteri greci . . . 369
18.7.3 Scrivere con altri alfabeti . . . 371
18.8 La gestione dei font . . . 371
18.8.1 Altri font già disponibili . . . 375
18.9 Installazione di altri font vettoriali . . . 377
18.9.1 Operazioni preliminari . . . 379
18.9.2 Installare un font Type 1 creato con FontForge . . . 381
18.9.3 Installare un font Type 1 creato con METAFONT . . . 382
18.9.4 Installare font Type 1 . . . 383
18.9.5 Installare font TrueType . . . 384
18.9.6 Installare font OpenType . . . 384
18.9.7 Aggiornamento dei file generali di mappa . . . 384
18.9.7.1 TeX Live . . . 384
18.9.7.2 MiKTeX . . . 386
18.9.8 I file di descrizione dei font . . . 387
18.9.9 Ricapitolazione . . . 392
18.9.10 Font e sistema TeX . . . 393
18.10 Conclusioni . . . 397

19 LaTeX: nuovi comandi **399**
19.1 Introduzione . . . 399
19.2 Le definizioni di comandi nuovi . . . 400
19.3 Ridefinizione di comandi già esistenti . . . 404
19.4 Ridefinizioni di comandi di sistema . . . 405
19.4.1 I numeri romani maiuscoletti . . . 405
19.4.2 La virgola intelligente . . . 410
19.4.3 Il numero nelle liste delle tabelle e delle figure . . . 415
19.5 Esiste già o non esiste ancora il comando? . . . 416
19.6 Definizione di comandi robusti . . . 417
19.7 Definizione di un nuovo ambiente . . . 418
19.8 La ridefinizione di ambienti esistenti . . . 420
19.9 Situazioni particolari . . . 424
19.9.1 Le linee guida . . . 424
19.9.2 Controllo della posizione di grandi oggetti . . . 427

19.9.3 Immagini, celle e scatole 428

20 LaTeX: la geometria delle pagine 437
20.1 Introduzione 437
20.2 La geometria della pagina 439
20.2.1 Il formato delle pagine 439
20.2.2 Le segnature e le imposizioni 441
20.2.3 I crocini 443
20.2.4 Dimensioni della gabbia del testo 444
20.2.4.1 Testatine e piedini; il pacchetto *fancyhdr* . . . 446
20.2.4.2 Le proporzioni della gabbia di testo 448
20.2.4.3 I margini 449
20.3 Lo scartamento e i contrografismi verticali 454
20.4 I capoversi 457
20.5 Testatine e piedini 459
20.6 I pacchetti di personalizzazione 460
20.6.1 Il pacchetto *geometry* 460
20.6.2 I pacchetti *titlesec*, *fancyhdr*, *sectsty* e *tocloft* 464
20.6.3 Testatine 466
20.7 La pagina del titolo 469
20.8 Le inserzioni flottanti e non flottanti 470
20.8.1 Gli oggetti flottanti 470
20.8.2 Gli oggetti non flottanti 471
20.9 Conclusioni 473

21 Dove documentarsi 475
21.1 La documentazione essenziale 475
21.2 Documentazione sulla tipografia 476
21.3 Documentazione su LaTeX 479
21.4 Documentazione sulla grafica 480
21.5 Documentazione sui singoli pacchetti 480
21.6 Documentazione su TeX 481
21.7 Documentazione sui simboli di LaTeX 482
21.8 Composizione della matematica 482
21.9 L'archiviazione dei documenti 484

22 Il formato PDF archiviabile 487
22.1 Le immagini 488
22.2 I font 490
22.3 Gli hyperlink 493
22.4 Generazione di un file PDF archiviabile 493
22.4.1 La strada maestra 493
22.4.2 Trasformazione di un file PDF o di un file PS 495
22.5 Conclusioni 498

23 Comporre documenti di molti autori **499**
23.1 Conversione manuale 500
23.1.1 Copia e incolla 501
23.2 Conversione automatica 507
23.2.1 Documenti in formato `.doc` 507
23.2.2 Documenti in formato PDF 510
23.2.3 Documenti in formato XML 512
23.3 Documenti in collaborazione 513
23.3.1 GIT 515
23.3.2 Organizzazione minimale 515
23.3.2.1 File `txt` 515
23.3.2.2 File MMD e MD 516
23.3.3 Andare oltre: Scrivener 517
23.3.3.1 Compilazione del documento finale in PDF . . 520
23.3.3.2 Compilazione del documento finale in file `tex` . 522
23.3.4 Commenti in stile *html* 522
23.4 Conclusioni 524

24 Simbologia matematica e fisica **527**
24.1 Unità di misura del Sistema Internazionale 527
24.2 Simboli matematici nelle scienze 530
24.3 Nomenclatura 532

25 Divisione in sillabe **557**
25.1 Quando viene eseguita la cesura 557
25.2 La sillabazione fonetica oppure etimologica 561
25.3 Come fa TEX a dividere in sillabe 562
25.4 Bruttezza residua 565
25.5 I pattern per la lingua italiana 566
25.6 Come rimediare a righe sporgenti dalla giustezza 568
25.7 I file di pattern 569

26 Codifica in entrata e codifica in uscita **575**
26.1 Introduzione 575
26.2 Le tre distinte codifiche di TEX 578
26.2.1 La codifica di ingresso 586
26.2.2 La codifica di uscita 588
26.3 Specificare la codifica giusta 590
26.3.1 Scoprire la codifica di input 592
26.3.2 Cambiamento della codifica 594
26.4 Collage di contributi diversi 598
26.5 Considerazioni riassuntive 598

27 Come fa TEX a comporre le pagine **601**
27.1 Divisione dei capoversi 602
27.2 Divisione della pagine 604
27.3 Cosa fare se... 606
27.3.1 Collocazione degli oggetti flottanti 606
27.3.2 Le equazioni ingombranti 607
27.3.3 I sezionamenti 608
27.4 Conclusioni 609

28 Trattamento degli errori **611**
28.1 Errori ortografici nei nomi dei comandi 611
28.2 Errore nella ricerca dei file 611
28.3 Ciclo infinito 613
28.4 Gruppi aperti 614
28.5 Mancata apertura di un gruppo 615
28.6 Interferenza con le definizioni 616
28.7 File personali che provocano conflitti 616
28.8 Tracciare l'operato del programma 618
28.8.1 I comandi primitivi di tracciamento 618
28.8.2 Il pacchetto *trace* 619
28.9 Costruzione dei capoversi e delle pagine 622
28.10 Conclusioni 625

29 Riepilogo della sintassi di LATEX **627**
29.1 La struttura del documento 629
29.2 Periodi e capoversi 630
29.2.1 Periodi 630
29.2.2 Capoversi 631
29.2.3 Note in calce 633
29.2.4 Note marginali 633
29.2.5 Accenti e simboli speciali 634
29.3 Suddivisione del testo e indici 634
29.3.1 Comandi di sezionamento 634
29.4 Classi, pacchetti e stili delle pagine 638
29.4.1 Classe del documento 638
29.4.2 Pacchetti 640
29.4.3 Stili delle pagine 641
29.4.4 Il frontespizio 643
29.5 Testi in display 645
29.5.1 Citazioni e poemi 645
29.5.2 Liste 645
29.5.3 Testo composto verbatim 650
29.6 Formule matematiche 651
29.6.1 Formule 651
29.6.2 Simboli, accenti, delimitatori e grandi operatori 654

29.6.3 Impilare gli oggetti matematici 654
29.6.4 Spaziatura matematica 654
29.6.5 Font matematici 655
29.6.6 Stili di composizione 656
29.7 Definizioni, numeri e programmazione 656
29.7.1 Comandi di definizione 656
29.7.2 Comandi per la definizione di ambienti 657
29.7.3 Teoremi 657
29.8 Numeri, lunghezze e spazi 658
29.8.1 Numeri 658
29.8.2 Lunghezze 662
29.8.3 Operazioni fra numeri e grandezze 663
29.8.4 Il pacchetti *ifthen* e *etoolbox* 667
29.8.4.1 Il pacchetto *ifthen* 667
29.8.4.2 Il pacchetto *etoolbox* 670
29.9 Spaziature 673
29.10 Figure, tabelle ed altri oggetti flottanti 674
29.10.1 Figure e tabelle 674
29.10.2 Note marginali 679
29.11 Incolonnamenti 681
29.11.1 L'ambiente tabbing 681
29.11.2 Gli ambienti array e tabular 682
29.12 I file ausiliari e i loro comandi 685
29.12.1 I file del sistema TEX 685
29.12.2 I riferimenti incrociati 686
29.12.3 Bibliografia e citazioni 686
29.12.4 Suddivisione del file sorgente 687
29.13 Indice analitico e glossario 688
29.13.1 Indice analitico 688
29.13.2 Glossario 689
29.14 Compilazione interattiva 689
29.15 Interruzione di riga e di pagina 690
29.15.1 Interruzione di riga 690
29.15.2 Interruzione di pagina 691
29.16 Scatole 692
29.16.1 Scatole di uso immediato 693
29.16.2 Scatole per conservare del testo 694
29.16.3 Ambienti e comandi per scatole particolari 695
29.16.4 I comandi per le scatole con il linguaggio di TEX 697
29.16.5 Operazioni di misura sulle scatole 700
29.17 Disegni e colori 701
29.17.1 Disegni 701
29.17.2 Colori e grafica 701
29.18 Selezione dei caratteri 704
29.18.1 Scegliere famiglia, forma e serie 704

29.18.2 Scegliere il corpo 705
29.18.3 Corpi testuali e matematici 705
29.18.4 Simboli speciali 706

Bibliografia **707**

Indice analitico **712**

Indice degli ambienti **728**

Indice delle classi **730**

Indice dei file **731**

Indice dei pacchetti **734**

Indice dei programmi e delle distribuzioni **737**

Elenco delle tabelle

4.1 Legame fra alcuni comandi di sistema, gli interpreti e i formati . 81

8.1 Descrittori delle colonne per le tabelle 156
8.2 Tabella la cui colonna di destra è specificata con `p{80mm}` . . . 158
8.3 Tabella adattata alla giustezza 163
8.4 Tabella male adattata alla giustezza 163
8.5 Tabella bene adattata alla giustezza 163
8.6 Tabella composta con le estensioni del pacchetto *array* 170

9.1 Tabella insolita 192
9.2 Esempio di valutazione di un seminario 200

10.1 Dimensioni di `texbook.pdf` con vari metodi 215

13.1 Le lettere greche 254
13.2 Gli operatori di relazione 255
13.3 Gli operatori binari 255
13.4 Gli operatori funzionali 256
13.5 I grandi operatori 256
13.6 I grandi delimitatori 257
13.7 Accenti e diacritici matematici 258
13.8 Altri simboli 258

14.1 Prima serie di simboli accessibili con il pacchetto *amsmath* . . . 270
14.2 Seconda serie di simboli accessibili con il pacchetto *amsmath* . 271
14.3 Gli ambienti di allineamento di *amsmath* 273

15.1 Formati britannici della pagina 298
15.2 I tasti della tastiera latina e i segni dell'alfabeto greco 302

18.1 Istruzioni per la scelta del corpo dei caratteri 353
18.2 I corpi ottici a confronto col corpo unico 354
18.3 Il font latino a 128 caratteri con codifica OT1 355
18.4 Il font cirillico a 128 caratteri con codifica OT2 355

18.5 Il font latino a 256 caratteri con codifica T1 356
18.6 Varie combinazioni di serie e forma per i caratteri standard . . . 358
18.7 Il corsivo matematico con codifica OML 362
18.8 La polizza dei simboli matematici con codifica OMS 362
18.9 I segni matematici estensibili con codifica OMX 363
18.10 Il Text Companion Font con codifica TS1 366
18.11 Il font cirillico esteso con la codifica *X2* 368
18.12 Il font greco a 256 caratteri con codifica LGR 370
18.13 Il font latino con codifica LY1 378

19.1 Allineamento scorretto in una tabella 432
19.2 Allineamento corretto in una tabella 432
19.3 Tabella con allineamento corretto e spaziatura corretta 432
19.4 Allineamento aggiustato in modo accurato 434

20.1 Formati delle pagine a seconda della segnatura 440
20.2 Giustezze determinate mediante la lunghezza dell'alfabeto . . . 447

24.1 Unità fondamentali 527
24.2 Prefissi decimali 528
24.3 Prefissi binari 528
24.4 Unità logaritmiche 528
24.5 Unità derivate 530
24.6 Unità di misura legalmente ammesse 531
24.7 Unità di misura temporaneamente accettate 532
24.8 Simboli matematici 533
24.9 Nomenclatura, simboli e unità di misura 547

25.1 Pattern usati per dividere in sillabe *dell'istruzione* 563
25.2 Pattern usati per dividere in sillabe *discinesia* 564

26.1 I 95 caratteri ASCII stampabili 578
26.2 Alcune codifiche di ingresso per i caratteri latini 582

29.1 Gli accenti testuali e i simboli speciali 635
29.2 Numeri associati al livello di sezionamento 636
29.3 Parametri nella classe `book` per gestire gli oggetti flottanti . . . 679
29.4 Dichiarazioni per la scelta di famiglia, serie e forma 705
29.5 Corrispondenza fra comandi e dichiarazioni 705
29.6 Dichiarazioni di corpo 705

Elenco delle figure

1.1 Semplice esempio di composizione con LaTeX. 4

2.1 Dedica a Donald Knuth 11
2.2 Auguri di TUG (TeX Users Group) per il 2003 11
2.3 Confronto relativo fra le varie scale di lunghezza 12
2.4 Caratteristiche dei caratteri mobili 13
2.5 Cassa dei caratteri di una tipografia italiana 14

4.1 Schermata di LaTeXiT 47
4.2 Schema della tastiera italiana sulle piattaforme Windows 56
4.3 Schema della tastiera configurata con il driver EurKey 58
4.4 Schema della tastiera svizzera sulle piattaforme Windows 59
4.5 Schema della tastiera italiana su Linux senza personalizzazioni . 60
4.6 Schema della tastiera USA estesa sul MacBook Pro 61

6.1 Tre geometrie di pagina a confronto 119
6.2 Proporzioni ISO e aurea della pagina 121

7.1 Il comando `\centering` e l'ambiente *center* 145

9.1 Relazione fra le dimensioni della figura e l'origine degli assi . . . 179
9.2 Le potenzialità dell'ambiente *picture* secondo Lamport 183
9.3 Curve di Bézier di secondo e terzo grado nell'ambiente *picture* . 183
9.4 Moto uniformemente accelerato 184
9.5 Disegno ottenuto con il pacchetto *pgf* e l'ambiente *tikzpicture* . 186
9.6 Disegno nell'ambiente *tikzpicture* usando il pacchetto *circuitikz* 187
9.7 Una paginetta con l'angolo ripiegato 188
9.8 Un esempio di figura geometrica composta con METAPOST . . . 190
9.9 Istogramma con il risultato della valutazione di un seminario . . 201
9.10 Diagramma con il risultato della valutazione di un seminario . . 202
9.11 Diagramma a torta con i tipi di produzione dell'energia elettrica 203

10.1 Confronto sugli effetti dell'ingrandimento su due figure simili . . 213
10.2 Inclusione e conversione di un file `eps` 217

10.3 Due foto trattate con diverse chiavi 229

12.1 Le fasi per la produzione dell'indice analitico 243

13.1 Due esempi di strafalcioni giornalistici 266

15.1 Sette contro Tebe 301
15.2 Esempio di composizione in greco 303
15.3 Alcuni alfabeti gotici che includono anche i capilettera ornati . 306
15.4 Completamento del file sorgente 316
15.5 Risultato della composizione con XeLaTeX 317

16.1 Otto slide per una presentazione di cinque frame 326

18.1 Il carattere metallico della 'm' 344
18.2 Il corpo, l'interlinea e l'avanzamento di riga 344
18.3 Composizione normale e sterlineata 345
18.4 Legature antiche 347
18.5 Il simbolo di *maschio* fortemente ingrandito 351
18.6 Esempi di testi composti con i font Times e Palatino 360
18.7 Il Credo di Nicea in tre lingue che fanno uso di alfabeti diversi . 372
18.8 I *dingbats* creati da Hermann Zapf 376
18.9 Schema grafico del processo di composizione 393
18.10 Presentazione all'esterno dei risultati della composizione 393
18.11 Processi di conversione dei vari formati di uscita 394
18.12 Trasformazioni di formato dei font e generazione dei file ausiliari 395
18.13 Gestione delle mappe dei font 396

19.1 Alcune proprietà delle scatole 429

20.1 Imposizioni per diverse segnature 444
20.2 Esempio di crocini predisposti per una tipografia italiana 445
20.3 Disegno della pagina secondo Gutenberg 450
20.4 Disegno della pagina con il metodo delle strisce 451
20.5 Confronto fra tre gabbie auree 453

23.1 La finestra di compilazione di Scrivener 519
23.2 La compilazione da Scrivener 521
23.3 La compilazione da Scrivener in MMD o LaTeX 523

Avvertenze

In questo testo si parla di quello che si può fare con la ricca dotazione di programmi e di pacchetti presenti in ogni distribuzione del sistema TEX. Le distribuzioni del sistema TEX gratuite sono essenzialmente due, TEX Live e MiKTEX. In commercio ci sono diverse altre distribuzioni.

Qui ci si riferisce principalmente alla distribuzione del sistema eseguita mediante il DVD prodotto dalla Associazione Internazionale degli Utenti di TEX (TEX Users Group, TUG) e ridistribuito anche ai membri dell'Associazione Italiana, GUIT (Gruppo degli Utenti Italiani di TEX e LATEX). Volendo, il DVD può venire acquistato ad un costo nominale di US$ 50,00 dal sito `http://www.tug.org"/store/`, ma nel sito di TUG si dice che l'immagine ISO del disco può essere scaricata gratuitamente. L'immagine ISO va poi masterizzata su un DVD vergine e usata tranquillamente alla pari del DVD originale. La distribuzione scaricata dal sito TUG menzionato è completa e permette l'installazione sulle macchine Windows, Linux e Macintosh.

Anche la distribuzione MiKTEX (valida solo per le macchine Windows) può venire scaricata gratuitamente dal sito `http://miktex.org/2.9/setup`. Più precisamente da questo sito è possibile scaricare dei semplici programmi di installazione, uno molto piccolo per installare una versione di base e uno di dimensioni maggiori per scaricare la distribuzione completa. Benché la distribuzione di base possa essere arricchita in modo praticamente automatico via via che l'utente richiama funzionalità ancora non installate, si consiglia vivamente di eseguire l'installazione completa.

Nel seguito si farà riferimento specialmente alla distribuzione TEX Live, perché con minime varianti è identica per tutte le piattaforme di elaborazione più comuni. Anzi, anche per chi lavora su macchine Windows, sebbene esista l'eccellente distribuzione MiKTEX, espressamente prodotta per lavorare con i vari sistemi operativi della Microsoft, si consiglia di installare la versione TEX Live. Certe personalizzazioni necessarie per la composizione tipografica di alta qualità diventano relativamente difficili da fare con la distribuzione MiKTEX, prevista per utenti ai quali, lavorando con un sistema operativo "proprietario", non sono concesse troppe personalizzazioni.

Nel seguito quando si scriverà CTAN ci si riferirà al Comprehensive TEX Archive Network (la rete degli archivi completi del sistema TEX); questa rete è formata da tre server principali dislocati in varie parti del mondo e da un numero sterminato di "mirror" che ne rispecchiano il contenuto. I primi sono aggiornati in modo sincrono ogni notte; i secondi sono aggiornati in modo dilazionato dipendente dalla loro politica amministrativa, ma generalmente sono molto aggiornati anche i mirror.

Nel seguito quando si userà l'indicazione `$TEXMF` ci si riferirà alla cartella o folder che funziona da "radice" per gli "alberi" di cartelle del sistema TEX. Questo sistema, infatti, ha i suoi file collocati in molte cartelle gerarchicamente collegate fra di loro in strutture ad albero; la radice di ciascuno di questi alberi è posta nel disco fisso dell'elaboratore in posizioni diverse a seconda della macchina

e del sistema operativo. Ogni utente conosce la sua macchina e sa dove trovare le cartelle radice dei suoi programmi preferiti.

Tuttavia, tranne dove indicato in modo esplicito, qui si userà solo il segno di barra diritta '/' come separatore fra i nomi delle cartelle; questo è il separatore normale per le macchine che lavorano con i sistemi operativi di tipo UNIX, incluso il sistema operativo delle macchine Macintosh recenti, etichettato con la sigla Mac OS X. Nelle macchine Windows il separatore è notoriamente la barra rovescia '\'; ma anche questo, all'interno dei file che contengono i testi da comporre con i programmi del sistema TEX va sostituito con la barra normale, poiché la barra rovescia per il sistema TEX ha un significato speciale.

I lettori ci vorranno perdonare se abbiamo dato la preferenza sia a TEX Live, sia alla barra diritta; questo è anche legato al fatto che come il sistema TEX fa parte del software libero, così abbiamo dato la preferenza ai sistemi operativi liberi o quasi. Ovviamente non c'è niente di male a servirsi di prodotti commerciali, che talvolta offrono qualcosa di più dei prodotti liberi. Anche nel campo delle distribuzioni del sistema TEX ci sono diverse distribuzioni commerciali che offrono qualcosa in più. Tutti i prodotti commerciali, però, danno qualcosa in meno, nel senso che l'utente è vincolato al prodotto e deve talvolta rinunciare alla possibilità di trasferire agevolmente programmi e documenti da una macchina ad un'altra con un diverso sistema operativo o priva dello stesso prodotto commerciale.

Una certa conoscenza preliminare del sistema TEX è utile per comprendere a fondo questo testo; in ogni caso si dà per scontato che l'utente conosca bene come lavorare sulla propria macchina di elaborazione e abbia una certa familiarità con l'uso della finestra chiamata "Terminale" o "Prompt dei comandi" nei vari sistemi operativi. Talvolta il sistema TEX richiede necessariamente di dare dei comandi in linea, scritti cioè in una di queste finestre specificatamente destinate ad interagire con la macchina mediante comandi scritti, senza usare il mouse o altri comandi "scorciatoia".

Capitolo 22

Il formato PDF archiviabile

In una guida come questa non può mancare un capitoletto sul formato PDF archiviabile. Questo formato soddisfa alla norma ISO 19005-1 del 2005 che stabilisce particolari restrizioni o specifiche al formato.

Come questa Guida nasce essenzialmente come libro elettronico (*e-book*), così una miriade di altri documenti nasce come documenti elettronici, che talvolta vengono stampati, ma sempre vengono archiviati. Come sarà possibile leggere questi documenti di archivio di qui a cinquanta anni? Ecco, questo è il motivo per il quale l'ISO ha provveduto e emettere le norme per l'archiviazione dei documenti elettronici.

Sostanzialmente il formato prescelto dall'ISO è il PDF (*Portable Document Format*) che già in partenza era nato come uno strumento relativamente aperto ma comunque indipendente dalla piattaforma sulla quale ogni documento era stato composto. La Adobe, che aveva sviluppato questo formato, aveva reso pubbliche le specifiche in modo che chiunque potesse scrivere programmi di conversione, di composizione o di visualizzazione per questo formato.

Gli Utenti di TEX ne sono perfettamente consapevoli, visto che usano pdftex nato in modo indipendente dalla Adobe; usano anche ghostscript, nato per leggere e stampare i documenti scritti in formato PostScript, e successivamente esteso a leggere e a scrivere (convertire) documenti in formato PDF. La collezione dei programmi del sistema TEX contiene anche il programma dvipdfm per convertire direttamente un file in formato DVI in un file corrispondente in formato PDF; lo stesso avviene per i vari altri programmi "fratelli' di pdftex: Aleph, ConTEXt, LuaTEX, eccetera. Nessuno di questi programmi è nato presso la Adobe, ma è stato possibile predisporli grazie al fatto che le specifiche del formato sono a disposizione di chiunque.

Era naturale che con queste premesse l'ISO si rivolgesse al formato PDF come base per il formato archiviabile. Essa però ha imposto che il documento archiviabile sia scritto con il linguaggio PDF nella versione 1.4 e che sia dotato di altre informazioni che consentano il reperimento dell'informazione utile con i sistemi archivistici più avanzati; per esempio, esso deve contenere dei *metadata*

che consentano come minimo di ricercare il documento sulla base del suo titolo e/o dei suoi autori; ancor meglio: questi *metadata* possono contenere anche le parole chiave per eseguire ricerche più mirate.

L'ISO impone anche che il documento sia indipendente dalle risorse dell'elaboratore sul quale viene eventualmente letto dopo molti anni dalla sua archiviazione; ovviamente l'elaboratore dovrà disporre di un lettore di documenti PDF, ma non necessariamente uno dei lettori esistenti oggi. Non basta: il documento deve essere autosufficiente anche per quel che riguarda i font con cui è composto, e deve completamente integrare ogni immagine della quale sia anche ben chiaro il modello di colore usato.

In sostanza i requisiti minimi richiesti ad un documento archiviabile, oltre ai *metadata* per le ricerche di archivio, sono che il documento sia leggibile esattamente nella stessa forma in cui era stato composto. Questi requisiti minimi configurano la specifica PDF/A-1b, meno restrittiva della specifica PDF/A-1a, la quale invece richiede che il file PDF sia anche un *tagged PDF* affinché sia estraibile anche la struttura logica del documento, indipendentemente dal fatto che questo contenga gli hyperlink interni per la lettura dinamica sullo schermo.

Ovviamente la Adobe ha provveduto con il suo programma Acrobat Professional, dalla versione 7.0 in poi, a rendere disponibile un certo numero di opzioni per trasformare file nati in formato PDF nei requisiti richiesti dalle norme ISO, sia per la variante PDF/A-1a sia per quella PDF/A-1b. Tra le opzioni c'è anche la possibilità di *verificare* se un file PDF prodotto esternamente sia già conforme all'una o all'altra delle due specifiche anzidette; eventualmente ne consente la correzione, in ogni caso ne segnala gli errori e le manchevolezze.

In commercio esistono diversi altri programmi di conversione e di verifica ma, a conoscenza dello scrivente e fino ad oggi (fine 2008), nessuno di quei programmi è disponibile gratuitamente, salvo, ovviamente, i programmi del sistema TEX. Infatti questo sistema contiene ora (dicembre 2008) le cose necessarie per produrre i file PDF/A-1b, ma è anche affiancato dal programma ghostscript (dalla versione 8.60 in poi) che riesce ad eseguire la conversione in modo diverso, ma conforme alle specifiche. Quello che il sistema TEX non contiene ancora è un programma di verifica della conformità, e l'utente non può fidarsi del fatto che, se l'esecuzione di pdflatex o di ghostscript è andata a buon fine, allora siano anche rispettate tutte le specifiche della norma ISO.

La produzione di un documento archiviabile non fa parte in senso stretto dell'Arte della Composizione Tipografica; tuttavia se un documento è stato prodotto con ogni cura e deve essere archiviato, è importante che il compositore segua anche quelle norme e quelle procedure che permettono di rispettare le norme ISO.

22.1 Le immagini

Le immagini incluse in un file archiviabile devono essere tutte conformi allo stesso modello di colore. Se si dispone di immagini in vero bianco e nero, questo può

essere accettato insieme agli altri modelli di colore, ma solitamente le immagini con così poca informazione (ovvero con un contrasto così alto) sono assai scadenti, quindi sarebbe meglio evitarle, specialmente se sono ottenute tramite uno scanner. È sempre possibile configurare lo scanner utilizzato per evitare l'uso di soli due bit per ogni pixel; sarebbe meglio configurarlo per un modello di colore a tonalità di grigio (8 bit per ogni pixel); meglio ancora è lasciare le impostazioni di default che si riferiscono in generale al modello di tricromia additiva RGB (*Red, Green, Blue*) che rendono perfettamente i colori (anche il nero) sugli schermi dove i puntini luminosi emettono luce corrispondente a quei tre colori fondamentali.

Talvolta si è portati a pensare alla stampa, la quale usa la quadricromia sottrattiva CMYK (*Cyan, Magenta, Yellow, blacK*), dove i puntini colorati sono ottenuti depositando particelle piccolissime di colore, che riflettono il colore desiderato assorbendo gli altri colori presenti nella luce bianca con cui viene illuminato lo stampato. È meglio non fare riferimento al modello CMYK; ogni stampante, a seconda del sistema di stampa che usa, del tipo di pigmenti colorati, eccetera, ha il suo convertitore di colore che trasforma il modello additivo RGB in quel che occorre alla stampante per stampare correttamente rendendo i colori al meglio.

Per cui tutte le immagini oltre a essere conformi allo stesso modello di colore, è bene che facciano riferimento al modello additivo RGB.

Questo implica che i *metadata* contenuti nel file PDF archiviabile menzionino esplicitamente il modello di colore. Ne deve specificare non solo il tipo, ma anche il file che lo descrive; questo tipo di file ha estensione `.icc`[1].

Se una ricerca sui file del proprio PC non presentasse nessun file con estensione `.icc`, allora bisogna scaricarsene uno di validità generale; si consiglia il sito dell'European Color Initiative (`http://www.eci.org`) dalla cui pagina di download è possibile scaricare il file `ECI-RGB.V1.0.icc` e fare riferimento a questo file su qualunque piattaforma. Ovviamente questo aspetto della configurazione andrà personalizzato da ciascun compositore secondo le sue necessità; quanto si è detto è sostanzialmente una indicazione generale, per giunta valida per tutti e tre i principali sistemi operativi.

Una attenzione particolare va posta nei modelli di colore e nei colori usati per i comandi interni di pdflatex; per esempio a chi scrive è successo di configurare con il colore *magenta* i link alle citazioni bibliografiche, constatando che il file non era conforme alle norme ISO; cambiando il colore in *blue* il file diventava conforme.

Questo succede perché in mancanza di configurazioni globali, i colori che si possono usare vengono tratti dai comandi interni che fanno uso del minimo di informazione necessaria; il colore *blue* è uno dei colori fondamentali del modello di colore RGB, mentre il *magenta* è uno dei colori fondamentali del modello di colore CMYK, e da questo viene tratta la codifica interna del colore da usare, chiaramente incompatibile con il modello RGB con cui era dichiarato il file e

[1] File un po' anzianotti potrebbero avere estensione `.icm`, ma sarebbe meglio evitarli e fare riferimento a file più recenti.

tutte le sue immagini. Basta usare correttamente i comandi di configurazione del pacchetto `color` o `xcolor` o `graphicx` e il problema svanisce anche con il colore *magenta* e gli altri colori fondamentali del modello CMYK.

22.2 I font

I file PDF archiviabili devono contenere tutti i font di cui fanno uso; essi devono risultare tutti *Embedded* quando se ne guardano le caratteristiche, per esempio, usando il programma gratuito Adobe Reader e chiedendogli di mostrare le proprietà del documento. In generale i font non vengono incorporati completamente, ma ne viene incorporato un sottoinsieme (*subset*, cosicché l'apposita finestra del Reader indicherà *Embedded subset*) per quei font di cui sono stati incorporati i soli caratteri effettivamente usati nel documento.

Ma attenzione: quando si incorpora una figura, anche PDF, che a sua volta contiene altri font, anche questi devono risultare incorporati. Ci si dimentica assai spesso che le figure prodotte con altri programmi e salvate in formato PDF, spesso non incorporano tutti i caratteri, in particolare i 35 font che si suppone esistano di default su qualunque macchina capace di visualizzare i file PostScript; oppure gli 11 font che si suppone esistano di default su qualunque macchina capace di visualizzare[2] i file PDF.

Bisogna quindi ricordarsi di configurare correttamente i programmi del sistema TEX, affinché incorporino *tutti* i font usati, anche quelli di default; ma bisogna ricordarsi di configurare anche i programmi di disegno affinché salvino le loro immagini in un formato che contenga almeno il sottoinsieme dei caratteri usati.

I programmi del sistema TEX sono configurati attraverso il file `updmap.cfg`; nelle prime, diciamo, 100 righe ci sono le impostazioni booleane per l'incorporazione dei font, sia per il programma dvips sia per i programmi pdftex e dvipdfm; per questi ultimi due è impostato il valore `true` per l'incorporazione dei font, mentre per il primo l'impostazione è `false`; quindi bisogna per prima cosa cambiare questo valore in `true` e poi rilanciare il programma eseguibile updmap al fine di creare le mappe dei font configurate correttamente.

Per le immagini incorporate o si riesce a configurare correttamente il programma di disegno usato, oppure bisogna ricorrere ad un trucco, non altrettanto efficace dal punto di vista estetico, ma accettabile. Il file PDF prodotto esternamente, può venire elaborato con il programma gratuito inkscape chiedendogli di salvare il risultato di nuovo in formato PDF; i font non incorporati, ma resi visibili attraverso i font in dotazione a questo programma, vengono sostituiti con i loro disegni vettoriali; il carattere vettoriale dell'immagine presente nel file originale viene conservato, ma dopo il trattamento il file ufficialmente non contiene più nessun font.

[2] Qui si usa la parola *visualizzare* anche quando il processo di visualizzazione consiste nella stampa.

Questo è il primo problema che si presenta con i font; il secondo problema è che ogni carattere deve avere una larghezza non nulla. Sembra ovvio, ma le cose non stanno così.

Infatti, specialmente con i font vettoriali (in formato `.pfb`) delle collezioni del sistema TEX, alcuni segni matematici vengono composti giustapponendo elementi che talvolta devono sovrapporsi ad altri elementi.

Fra i font che presentano questo *difetto* ci sono tutti i font matematici `cmsy` e i loro parenti di altre collezioni derivati dai font Computer Modern. I segni problematici sono `\not` e `\mapsto`. La presenza anche di una sola istanza di uno di questi segni in un file altrimenti ineccepibile, rende il file PDF non conforme alle norme ISO e non è archiviabile, anche se a schermo esso appare perfetto.

Nella descrizione del pacchetto *`pdfx`*, di cui si parlerà fra non molto, e nel sito `http://support.river-valley.com/wiki/index.php?title=Generating_PDF/A_compliant_PDFs_from_pdftex`, vengono suggeriti alcuni modi di aggiustare la situazione; chiaramente il lettore può seguire quei consigli; qui si presenta una soluzione alternativa, non esattamente equivalente, ma di validità generale e che non richiede la "manomissione" dei file appartenenti al sistema TEX; se e quando i curatori di quelle collezioni di font vorranno provvedere ad eseguire nativamente le correzioni necessarie, non occorrerà più ricorrere a nessun aggiustamento.

Il comando `\not` viene sovrapposto a qualunque operatore matematico di relazione per negarlo: tipicamente `\not=`, o il suo equivalente `\neq`, produce il segno $\neq$, ottenuto sovrapponendo il segno / al segno $=$. Ai tempi in cui nacque il sistema TEX le memorie RAM e i clock che regolavano l'esecuzione delle istruzioni macchina anche nei grossi *main frame* erano rispettivamente assai modeste e assai lenti; risparmiare su qualunque cosa era importante, per cui i segni matematici di relazione negati erano tutti ottenuti in questo modo; i font dell'American Mathematical Society hanno rimediato al brutto aspetto di alcuni di questi segni negati, ma il meccanismo è rimasto, anche per compatibilità con i milioni di documenti scritti in passato, anche oggi che sono passati 30 anni dalla nascita del sistema.

Purtroppo il segno `\not` privo di dimensioni orizzontali non è accettabile dalle norme ISO, per cui bisogna provvedere a sostituirlo con un trucco di programmazione base di TEX, consistente nel comando primitivo `\mathchoice`; questo comando per ciascuna delle quattro modalità di composizione della matematica (in display, in linea con il testo, indici primi, indici secondi) offre quattro modalità distinte di operazione che vengono specificate ordinatamente nei suoi quattro argomenti. La "correzione" ricorre a comporre dentro una scatola di larghezza nulla, una barra di dimensioni orizzontali non nulle, specificandone una dimensione diversa per ciascuno dei quattro modi. Ecco il codice, dove si sono volutamente incolonnate le corrispondenti graffe aperte e chiuse in modo da rendersi conto più facilmente delle quattro scelte:

```
\renewcommand*\not{%
\mathrel{%
   \mathchoice%
```

```
{\rlap{$\displaystyle\mkern2.5mu\mathnormal{/}$}}%
{\rlap{$\textstyle\mkern2.5mu\mathnormal{/}$}}%
{\rlap{$\scriptstyle\mkern2.5mu\mathnormal{/}$}}%
{\rlap{$\scriptscriptstyle\mkern2.5mu\mathnormal{/}$}}%
  }%
            }%
```

Il comando `\rlap` (*right overlap*) serve per comporre una scatola di larghezza nulla il cui contenuto si sovrappone a qualunque cosa si trovi alla sua destra; siccome al suo interno il modo di composizione torna ad essere quello testuale, bisogna riattivare il modo matematico e al suo interno, per ciascuna delle quattro modalità di composizione, bisogna specificare lo stile di composizione rispettivamente mediante i comandi `\displaystyle`, `\textstyle`, `\scriptstyle` e `\scriptscriptstyle`. Il comando `\mathrel` serve a specificare che il comando `\not` è un operatore di relazione, mentre `\mathnormal` serve per scegliere il font matematico da cui trarre la barra.

Il comando `\mapsto` di per sé non sarebbe problematico e produce il segno $\mapsto$. Il problema nasce dalla sua coda ‣ che è definita con una dimensione orizzontale nulla; non è chiaro il motivo, in quanto esso compare solo nella definizione di `\mapsto`; sarebbe stato forse preferibile disegnare il segno direttamente senza ricorrere alla coda di larghezza nulla aggiunta ad una freccia normale; chissà, forse nelle prime versioni del sistema TEX questo segno serviva anche ad altro. Il fatto è che esso non è accettabile in un file conforme allo standard PDF/A.

In questo caso si è preferito usare i comando `\rule` di LATEX; ahimè, questa soluzione è usabile solo con LATEX e ne è escluso l'uso con TEX[3]. Tuttavia... questa Guida si riferisce all'uso di LATEX per comporre tipograficamente in modo eccellente. La soluzione trovata ricorre ancora a `\mathchoice` ma il segno è ottenuto con dimensioni diverse per i quattro modi di composizione distinti:

```
\renewcommand\mapstochar{\mathrel{\mathchoice
{\rlap{\rule[.05ex]{.1ex}{1ex}}\mkern-.5mu}%
{\rlap{\rule[.05ex]{.1ex}{1ex}}\mkern-.5mu}%
{\rlap{\rule[.035ex]{.08ex}{.75ex}}\mkern-.5mu}%
{\rlap{\rule[.025ex]{.065ex}{.55ex}}\mkern-.5mu}%
}}
```

I numeri "magici" usati servono per legare lo spessore del segno e la posizione precisa a seconda dello stile di composizione matematico; sono tutti valori ricavati sperimentalmente e, ovviamente, non si tratta di prescrizioni immutabili, ma vanno adattate anche a font diversi da quelli qui ipotizzati dei simboli matematici della collezione Computer Modern.

Queste due soluzioni non toccano i font originali; possono essere inserite in un file di opzioni personali da usare sempre, o soltanto quando si devono produrre file PDF archiviabili. Speriamo solo che in un lasso di tempo ragionevole

[3] O meglio, il comando `\rule` di LATEX deve venire sostituito dal comando `\vrule` di TEX corredandolo delle necessarie specificazioni di altezza larghezza e profondità.

queste correzioni, o meglio correzioni più professionali vengano apportate ai font originali e vengano ridefiniti i comandi di generazione di questi segni, in modo che il risultato finale sia lo stesso, ma sia anche compatibile con il formato PDF archiviabile.

Al contrario, potrebbe succedere che altri file che fanno uso di altri font possano dare luogo a problemi simili; a seconda della diagnosi eseguita (oggi) con Acrobat Professional, ci si comporterà adeguatamente, ma in modo analogo a quanto descritto sopra.

22.3 Gli hyperlink

Anche gli hyperlink possono dare dei problemi, ma il pacchetto *hyperref* accetta ora la nuova opzione *pdfa* che consente di evitare questi problemi. L'unico che resta è quello di ricordarsi di specificare l'opzione. A questo scopo il pacchetto *pdfx* provvede da solo; ma se per qualche motivo non fosse possibile usare pdflatex, non si potrebbe usare *pdfx* e bisognerebbe passare attraverso il formato PostScript e a ghostscript; in questo caso bisogna ricordarsi di specificare bene l'opzione alla chiamata esplicita di *hyperref*.

Vale la pena di ricordare che nel formato PDF/A gli hyperlink interni continuano ad essere attivi, ma gli hyperlink verso indirizzi esterni, pur restando colorati come qualunque altro hyperlink, sono disattivati; la ratio di questa limitazione è che già oggi si incontrano molti hyperlink che non funzionano più; probabilmente fra cinquanta anni nessuno degli indirizzi internet oggi esistenti sarà più valido. Sarebbe quindi perfettamente inutile indirizzare i lettori verso indirizzi 'morti'. Bisogna tenere conto di questo fatto, specialmente quando si scrivono i riferimenti nella bibliografia o nel testo, in modo da evitare riferimenti ad indirizzi internet esterni.

22.4 Generazione di un file PDF archiviabile

Affrontati i problemi a cui bisogna porre attenzione (che riflettono un po' anche la strada tortuosa che chi scrive ha dovuto scoprire "sperimentalmente", visto che la documentazione era e resta (per ora) spiacevolmente carente) non resta che procedere alla generazione di un documento in formato PDF archiviabile.

22.4.1 La strada maestra

La strada maestra è quella di usare pdflatex richiamando il pacchetto *pdfx*. Questo pacchetto è recentissimo; se ne suggerisce l'installazione e la lettura attenta della sua documentazione, tenendo conto anche di ciò che è detto qui.

Il pacchetto, solo in congiunzione con con pdflatex, serve per produrre file in formato PDF/A-1b se gli specifica l'opzione *a-1b* oppure file in formato PDF/X-1a se gli specifica l'opzione *x-1a*; il formato PDF/A-1b è quello che ci interessa qui. Il formato PDF/X-1a serve per rispettare un'altra norma ISO

indirizzata alla conformità dei file stampabili; le problematiche sono simili, ma i file stampabili sono completamente orientati al modello di colore CMYK e dispongono di una serie differente di *metadata*.

Siccome il pacchetto *pdfx* carica a sua volta il pacchetto *hyperref* con le giuste opzioni a seconda dei formati, esso, *pdfx*, deve essere invocato per ultimo nel preambolo del documento archiviabile, o per ultimo nel file di classe del documento; ne consegue che eventuali personalizzazioni dei comandi relativi all'aspetto degli hyperlink devono venire eseguiti subito dopo la chiamata di *pdfx*, vale a dire alla fine del preambolo, se il pacchetto viene invocato per ultimo nel preambolo, oppure all'inizio del preambolo, se la chiamata del pacchetto viene eseguita alla fine del file di classe.

Bisogna però, almeno la prima volta, configurare il file `pdfx.sty` in modo da invocare il modello di colore giusto, a meno che non si voglia usare quello di default. Contrariamente al solito è esplicitamente consentito personalizzare il file `pdfx.sty`, ma a causa di ciò esso e i suoi file accessori devono essere montati in una cartella dell'albero personale, altrimenti al primo aggiornamento del sistema TEX ogni personalizzazione viene persa. Però, se sulla stessa macchina operano diverse persone, ognuna ha il suo albero personale e ognuna è libera di usare il modello di colore che vuole.

Si è suggerito di caricare il modello di colore descritto mediante il file `ECI-RGB.V1.0.icc`; perché il pacchetto *pdfx* lo possa trovare ed usare, è necessario che esso si trovi sul percorso di ricerca; il posto migliore è proprio la stessa cartella dove si trova il pacchetto. Ci si ricordi, lo si ripete, di personalizzare il pacchetto *pdfx* scrivendo il nome da solo o il nome con l'estensione nei punti dove il file originale `pdfx.sty` lo richiede.

Il punto essenziale che invece non bisogna dimenticare serve per impostare i *metadata* relativi al documento da archiviare. Questo va fatto evidentemente per ogni documento da archiviare e sempre con dati diversi. La cosa migliore da fare è quella di scrivere ogni volta un file con estensione `.xmpdata` e con il nome coincidente con il file principale (o unico) che contiene il testo sorgente `.tex` da comporre. Nel caso di questa Guida, per esempio, il cui file principale si chiama `GuidaGuIT.tex`, bisogna predisporre un file dal nome `GuidaGuIT.xmpdata` e con il contenuto corrispondente alla sintassi seguente:

```
\Title{⟨Titolo del documento⟩}
\Author{⟨Lista degli autori⟩}
\Keywords{⟨lista delle parole chiave⟩}
\Org{⟨Organizzazione cui appartengono gli autori⟩}
```

Si possono inserire anche altri comandi (tutti inizianti con una lettera maiuscola) e se ne può vedere l'elenco completo nella documentazione del pacchetto *pdfx* (oltre che nel suo listato, sempre molto istruttivo).

Non c'è molto da dire sul significato degli argomenti; si noti che la lista delle parole chiave sarebbe meglio che fosse una lista strutturata, non semplicemente una lista di parole separate da virgole; tenuto conto che l'apertura e la chiusura

della struttura sono già inserite di default, quello che serve è solo mettere i delimitatori fra una parola chiave e l'altra; la sintassi è la seguente:

⟨*prima chiave*⟩`</rdf:li><rdf:li>%`
⟨*seconda chiave*⟩`</rdf:li><rdf:li>%`
...
⟨*penultima chiave*⟩`</rdf:li><rdf:li>%`
⟨*ultima chiave*⟩

Attenzione: qui si è indicato con il segno di commento `%` dove la riga è spezzata, ma sarebbe meglio non andare mai a capo e scrivere tutte le parole di seguito; diventerà una riga lunga, quella delle parole chiave, ma si è più sicuri di non introdurre caratteri spuri; ricordiamo che quanto si scrive qui, in una forma praticamente identica va a finire nella sezione dei *metadata* del file PDF e quindi deve rispettare scrupolosamente le convenzioni del mark-up chiamato XMP, che è un sottoinsieme del linguaggio XML. Chi conosce come manovrare con il mark-up di LaTeX non necessariamente padroneggia anche il mark-up XML.

Fatto questo bisogna eseguire pdflatex quante volte occorre per soddisfare anche le esigenze della bibliografia e dell'indice analitico e poi bisogna sottoporre a verifica di conformità il file ottenuto. Scaramanticamente sarebbe opportuno incrociare le dita. Non si dispone ancora di abbastanza esperienza per sapere se si è riusciti ad evitare tutti i trabocchetti con i *metadata*, le immagini, i colori e i font che potrebbero produrre un file non conforme. Tuttavia chi scrive ha già prodotto diversi file archiviabili e ormai ha sviluppato una piccola esperienza che gli permette di affermare che se si seguono le procedure descritte sopra, le possibilità di incontrare problemi di conformità sono ridotte; in ogni caso si è indicata la strada per affrontare alcuni problemi con i font e con i file inclusi.

22.4.2 Trasformazione di un file PDF o di un file PS

Il programma ghostscript permette di convertire un file PS in un file PDF e l'uno e l'altro in file conformi agli standard PDF/A-1b oppure PDF/X-1a. Attenzione: questa possibilità di conversione si ha dalla versione 8.60 in poi.

La documentazione di ghostscript relativa a queste trasformazioni lascerebbe intendere che si possa usare il programma ps2pdf, ma chi scrive ha sperimentato che non è propriamente così; in ogni caso, il lavoro vero lo fa ghostscript e il risultato può essere conforme agli standard con la stessa probabilità che si ha usando pdflatex.

Per non dimenticare per la strada uno qualsiasi dei comandi o delle opzioni o dei file che ghostscript richiede, è meglio confezionarsi un comando mediante un file `.bat` per le macchine Windows oppure uno script bash per le macchine UNIX/Linux/Mac OS X.

Per le macchine Windows il file per trasformare i file PDF in file PFD/A potrebbe chiamarsi `pdf2pdfa.bat`, dovrebbe trovarsi sul percorso di ricerca dei file, e dovrebbe contenere quanto segue:

```
@echo off
rem procedura batch per lanciare gswin32c per trasformare
rem un file PDF in un file PDFA
rem QUANTO SEGUE DEVE ESSERE SCRITTO SU UNA SOLA RIGA!
gswin32c -dPDFA  -dNOOUTERSAVE -dUseCIEColor -sDEVICE=pdfwrite
-sProcessColorModel=DeviceRGB -o %1-a.pdf ./PDFA_def.ps %1.pdf
```

Per trasformare un file PS basta scrivere un'altra procedura identica a quella appena illustrata, ma con il nome, per esempio, `ps2pdfa.bat`, dove si usa l'estensione `.ps` nell'ultimo elemento del lungo comando.

Per le macchine UNIX/Linux/Mac OS X lo script bash potrebbe essere contenuto in un file al quale si potrebbe dare il nome `pdf2pdfa` senza scordarsi di dichiararlo eseguibile (`chmod 755 pdf2pdfa`) e di collocarlo in una cartella che si trovi sul percorso di ricerca; il suo contenuto non è molto diverso da quello scritto per le piattaforme Windows; semplicemente sfrutta la migliore sintassi del linguaggio di scripting bash; esso è il seguente:

```
#!/bin/bash
file1=$1.pdf
file2=$1-a.pdf
# QUANTO SEGUE DEVE STARE SU UNA SOLA RIGA
gs -dPDFA  -dNOOUTERSAVE -dUseCIEColor -sDEVICE=pdfwrite
-sProcessColorModel=DeviceRGB -o $file2 ./PDFA_def.ps $file1
```

Per trasformare un file PS basta creare uno script del tutto simile, chiamato per esempio `ps2pdfa`, dove si è sostituita con l'estensione `.ps` quella del file che serve per definire la variabile `file1`.

L'uso dell'una o dell'altra procedura è molto semplice: basta usare la finestra comandi e dare il comando:

`pdf2pdfa` ⟨*nome file senza estensione*⟩

Ma non basta: prima di mandare in esecuzione il programma ghostscript per eseguire la trasformazione, bisogna aver predisposto un file `PDA_def.ps` nella cartella dove risiede il file da convertire. Il contenuto di questo file è duplice: serve per definire il `DeviceRGB` e serve per introdurre i *metadata* necessari. Basta prendere il file con lo stesso nome dalle cartelle dove si trovano i file che fanno parte della distribuzione di ghostscript, copiarlo nella cartella che interessa il file da trasformare, e usarlo come *template* per inserire le informazioni necessarie.

Ci sono tre righe che contengono il commento `% Customize`. Queste righe sono da modificare o da ampliare secondo le proprie necessità. Attenzione: il file in questione è un file di puro testo, ma ha l'estensione `.ps`; questa estensione di solito è associata a programmi di resa grafica dei file PostScript; in questo caso si tratta di un file di puro testo, scritto in linguaggio PostScript, e deve venire aperto con un editor che non faccia danni: va benissimo l'editor con il quale normalmente vengono scritti i file sorgente `.tex`.

Le righe che devono essere personalizzate, devono venire modificate così:

- La riga che contiene il testo

```
/ICCProfile (ISO Coated sb.icc) % Customize.
```

va commentata premettendovi un segno % e subito dopo ne va inserita un'altra adeguata:

```
%/ICCProfile (ISO Coated sb.icc) % Customize.
% per macchine Windows
/ICCProfile (C:\icc\ECI-RGB.V1.0.icc)
%
% oppure
%
% per macchine UNIX/Linux/Mac OS X
/ICCProfile (~/icc/ECI-RGB.V1.0.icc)
```

Si è supposto, a titolo di esempio, che il file sia stato salvato in una cartella di nome `icc`, ma non è importante dove sia conservato il file purché nel precedente comando vi sia il percorso completo a partire, se non dalla radice del disco, almeno dalla propria "Home", concetto che è ormai acquisito anche nelle macchine Windows, anche se il suo percorso può essere assai diverso da quello delle macchine UNIX o simili. Ovviamente nella personalizzazione indicata sopra si userà la sintassi per macchine Windows o per le altre macchine a seconda di quale macchina si sta usando.

- La riga che contiene il testo

  ```
  [ /Title (Title) % Customize.
  ```

  va adeguata al file specifico che si sta convertendo; per esempio, per trasformare il file di questa Guida si possono usare le righe seguenti:

  ```
  [ /Title (Introduzione all'arte della composizione
            tipografica con LaTeX)
  /Author (Gruppo Utilizzatori Italiani di TeX e LaTeX)
  /Subject (Manuale di istruzioni per usare LaTeX
            consapevolmente)
  ```

  Si possono inserire altre informazioni, per esempio una lista di parole chiave per facilitare le ricerche di archivio; ma quelle relative al titolo e all'autore sono il minimo necessario.

- Il terzo punto da personalizzare è quello che corrisponde alla riga:

  ```
  /OutputConditionIdentifier (CGATS TR001) % Customize
  ```

  ma qui non è necessario intervenire, se non per inserire altre informazioni di cui si conosce la sintassi e il significato. Probabilmente questa riga riguarda la trasformazione nel formato PDF/X, ma non interferisce con la trasformazione PDF/A.

Con queste personalizzazioni e con le procedure indicate sopra la conversione procede spedita e con una ragionevole probabilità di successo; i problemi che si possono presentare in sede di verifica sono gli stessi, sostanzialmente, di quelli che si presentano lavorando con pdflatex.

22.5 Conclusioni

Il lettore non si scoraggi per la relativa rusticità delle procedure per la composizione diretta o per la trasformazione di file esistenti in file conformi alla specifica ISO19005-1:2005.

Il problema è difficile e oggi (fine 2008) siamo appena gli inizi della necessità di archiviare documenti in forma conforme alla specifiche normative.

Si tenga presente che gli stessi problemi si pongono con la suite Microsoft Office 2007 e con OpenOffice.org nella versione 3.x; persino il programma principe della Adobe, Acrobat Professional versione 8.x, incontra i suoi problemi.

A chi scrive (e che si è già fatto una piccola esperienza in merito) appare più corretto non nascondere le difficoltà e i problemi, che non proporre agli utenti, spesso acquirenti che sborsano cifre non trascurabili, programmi che non possono fare più di tanto e che non centrano il bersaglio più spesso di quanto non si creda.

Dispiace che il bersaglio non sia sempre centrato nemmeno con OpenOffice.org, che, se non altro, ha due pregi: (*a*) fin dai suoi inizi ha usato come formato di default quello con il mark-up XML, e (*b*) si tratta di un programma gratuito in mano alla comunità degli utenti, non molto diversamente da quel che succede con il sistema TEX, pur restando su un terreno ben diverso[4]. È quindi un peccato che anche OpenOffice.org sia scivolato su questa questione, senza una parola di avviso relativa alla difficoltà di salvare file veramente conformi alla norma ISO.

[4]La matematica è sempre stato un problema difficile da affrontare con programmi di tipo WYSIWYG; OpenOffice.org ha però un plug-in che consente di comporre le formule con LATEX; è sulla buona strada, ma se uno deve avere sulla propria macchina anche il sistema TEX per comporre le formule di OpenOffice.org, a che cosa serve usare OpenOffice.org? Solo per poter usare la tecnica WYSIWYG? Non so se la notizia sia vera, ma sembra che il nostro Grand Wizard, Donald E. Knuth, abbia collaborato con la Microsoft per l'implementazione del nuovo *equation editor* della versione 2007 di MS Office. Ultima notizia, ancora semiufficiale, reperibile in merito alla descrizione del plug-in Writer2TeX di OpenOffice.org: ora è ancora nella versione beta, ma è già scaricabile dal sito questo plug-in che consentirebbe di usare la componente Writer di OpenOffice.org come editor quasi WYSIWYG per produrre direttamente un file con il mark-up di LATEX, con i bottoni per eseguire la composizione sia con latex sia con pdflatex, e poi di eseguirne la visualizzazione su schermo: forse Writer vuole sostituire LyX? Può darsi, e tutto sommato non sarebbe un male, purché il file LATEX prodotto sia un file efficace ed efficiente. Per ora (giugno 2010) la versione beta di questo plug-in non è ancora completamente affidabile. È difficile dire se questa soluzione sia un vantaggio per gli utenti di LATEX, ma il futuro ce lo dirà. Si veda quanto esposto più avanti nel capitolo 23.

Capitolo 23

Comporre documenti di molti autori

Un problema che sorge spesso consiste nel comporre un'opera collettiva dove i vari autori scrivono usando programmi diversi; del tutto simile è la situazione di colui che si trova nella difficile situazione di dover comporre un testo scritto da un altro autore che ha usato uno strumento "scrittorio" diverso da LaTeX.

Occorre convertire il formato dei testi sorgente per introdurli nell'unico master file del documento LaTeX o, più frequentemente, per creare nuovi file da far leggere al master file.

L'evento più frequente è quello che richiede di aggiungere il mark-up di LaTeX in un file relativo ad un documento scritto inizialmente con un word processor.

Naturalmente potrebbe anche succedere di dover fare conversioni opposte, da LaTeX ad un altro formato. Qui non si tratteranno queste conversioni; ci si limita a citare il programma TeX4ht, generalmente facente parte di ogni distribuzione del sistema TeX non minimale o di base. Questo programma può convertire nel formato HTML un documento scritto con il mark-up della versione standard di LaTeX, cioè che non faccia uso di pacchetti di estensione generici, in particolare che non faccia uso di macro personali del compositore. Chi l'ha usato ne dice un gran bene, ma sottolinea la limitazione fondamentale che il file da convertire deve essere strettamente conforme al linguaggio LaTeX di base, qual è descritto nella guida di Lamport [35].

Un programma molto versatile sembra essere Pandoc, [17]. Questo programma, da usare dal terminale, permette di convertire ogni documento scritto virtualmente in qualunque formato in un altro documento scritto, sempre virtualmente, in qualunque altro formato. In realtà i formati in entrata e i formati in uscita non sono tutti i possibili formati, ma le due liste, benché diverse, sono talmente ricche che premettere 'virtualmente' alla parola 'qualunque' non sembra fuori luogo. Mediante opportune opzioni è possibile configurare l'esecuzione del programma per eseguire la conversione in modalità diverse. Fra le altre possi-

bilità esiste la possibilità di convertire un file LaTeX in un file XML, HTML e anche EPUB (sia pure con certe limitazioni). Si rinvia il lettore alla pagina web `http://johnmacfarlane.net/pandoc/` da dove è possibile scaricare sia il programma sia il manuale d'uso.

D'altra parte questa Introduzione tratta della composizione tipografica con LaTeX, quindi appare del tutto ragionevole esaminare come si possa creare un documento compilabile con `pdflatex` partendo da una varietà di documenti di varia origine.

23.1 Conversione manuale

La conversione manuale si può dire che richieda non poca fatica, ma è applicabile qualunque sia l'origine dei documenti da comporre e da assemblare assieme in una collettanea. Di solito si tratta di documenti scritti facendo uso di un word processor come Microsoft Word, StarOffice Writer o il Writer della collezione OpenOffice.org (fratello di StarOffice, ma con le caratteristiche del software Open Source) o del fork LibreOffice; potrebbero essere documenti in formato PDF oppure documenti in rete in formato HTML; potrebbero essere scansioni di documenti stampati; potrebbero essere persino dei manoscritti, nel senso letterale del termine: documenti scritti a mano! Insomma, il formato delle informazioni da elaborare può essere veramente eterogeneo.

Mentre un documento Word o Writer potrebbe venire convertito facendo uso di opportuni programmi, gli altri formati potrebbero essere difficili da elaborare in modo automatico; esistono diversi strumenti che consentono di eseguire alcune operazioni preliminari, che facilitano non poco l'elaborazione manuale.

Per i documenti HTML o si toglie il mark-up specifico di quel formato, oppure bisogna convertire il file HTML in un altro file con il formato XML o anche `.doc` e poi procedere con i programmi che verranno esaminati nei prossimi paragrafi.

I documenti scanditi[1] possono essere elaborati attraverso un apparato e un programma di Optical Character Recognition (OCR); sostanzialmente si scandisce di nuovo una stampa del file grafico, frutto della prima scansione, ordinando allo scanner di eseguire il riconoscimento dei caratteri; il risultato può essere molto buono o decisamente deludente; dipende da quanto era stata accurata la prima scansione e dipende molto dal tipo di caratteri usati; il riconoscimento automatico riesce meglio con certi caratteri piuttosto che con altri. In ogni caso il file in formato `.txt` (da preferire, anche se lo scanner può produrre anche un file in formato `.doc`) contiene molti errori di ortografia, dovuti al mancato

[1] Questo libro non è certo un modello di purezza linguistica; tuttavia il curatore si rifiuta di usare *scannerizzare* oppure *scansionare*, benché siano i termini più usati. Il verbo *scandire* e il sostantivo che ne deriva, *scansione*, rendono in italiano esattamente la stessa idea. Il nome *scansione* era già usato 50-60 anni fa per descrivere il movimento del pennello elettronico nei cinescopi per eccitare sequenzialmente tutte la particelle fotoluminescenti che avrebbero prodotto l'immagine televisiva. È esattamente il movimento del raggio luminoso che illumina sequenzialmente l'immagine da scandire nell'apparecchio che ha il nome tecnico inglese di *scanner*, ma per il quale il sostantivo italiano *scanditore* farebbe ridere tutti.

riconoscimento di certi caratteri, e quindi richiede di essere rivisto e corretto con attenzione.

Le immagini presenti nella scansione originale possono solitamente venire estratte, sperando di non degradare troppo la loro qualità, altrimenti devono venire eseguite *ex novo* tutte le immagini costituite da disegni al tratto; mentre per le fotografie o si può accedere agli originali, oppure bisogna accontentarsi. Questo è un problema comune anche quando si deve trasformare un file originario in formato `.doc`.

Comunque, ultimati questi preparativi possiamo supporre di avere a disposizione il testo da trasformare in formato di testo semplice oppure di testo formattato, contenente eventualmente immagini, tabelle, formule, cambiamenti di stile dei font, cambiamenti di margini, e simili; il testo, insomma potrebbe essere in formato `.txt`, oppure `.doc` o `.rtf`, oppure `.pdf`; sì, anche in formato PDF; ogni visualizzatore di questo formato, anche il semplice Adobe Reader, permette di selezionare il testo e di copiarlo per incollarlo altrove; bisogna semplicemente scegliere dalla barra superiore lo "strumento" per selezionare il testo, generalmente caratterizzato da una icona a forma di **T**.

Qualunque metodo si usi per trasformare il file, bisogna ricordarsi che un documento puramente testuale non contiene nessuna informazione descrittiva del testo; un file formattato, oltre al testo, contiene informazioni solamente riguardanti il modo di apparire, non quello che esso rappresenta veramente; il suo modesto mark-up contiene solo poche informazioni sulle immagini, le tabelle, le formule, i font, ma raramente contiene informazioni sul tipo di contenuto di un brano di testo, diversamente da come fa LaTeX. Questo significa che queste informazioni vanno aggiunte a mano dalla persona che si occupa della trasformazione, anche se per alcune parti di questa operazione si avvale di strumenti automatici; questi faranno il possibile, ma certamente non potranno aggiungere informazioni che dipendono dalla comprensione del testo se non sono comprese nel mark-up interno del documento. Bisogna ricordarsene anche quando si discuterà della conversione "semiautomatica".

Ciò premesso, e ben consci del lavoro necessario per una buona conversione, vediamo come eseguire la conversione completamente manuale.

23.1.1 Copia e incolla

Sembra semplice dire che si esegue la trasformazione manuale con la tecnica del copia e incolla; concettualmente non c'è nient'altro da fare, ma ci sono alcune operazioni che conviene tenere presenti.

1. Bisogna predisporre o un master document, o, comunque, una intelaiatura del file LaTeX nel quale avverrà la conversione; dalla dichiarazione della classe alla specificazione dei pacchetti che si intende o si prevede di dover usare, fino all'intelaiatura dell'ambiente *document*, da riempire via via con il testo da comporre. Se occorressero altri pacchetti, li si può sempre invocare

in un secondo tempo, ma è meglio aver studiato prima il documento da convertire per farsi un piano di lavoro destinato agli interventi da eseguire.

2. Conviene procedere un capoverso alla volta; lo si copia dalla finestra dell'applicativo che consente di visualizzare il formato del documento da convertire, e lo si incolla nella finestra di input del file LaTeX. Se c'è qualche intervento da fare, lo si esegue subito: per esempio per mettere in enfasi alcune parole, introdurre le virgolette giuste o i caporali giusti, eventualmente cambiando le virgolette copiate (vedi più avanti i cambiamenti necessari per i caratteri non ASCII); eseguire l'impostazione delle formulette matematiche in linea, e simili piccoli interventi che, proprio perché piccoli, sfuggirebbero se si copiasse l'intero documento in un colpo solo.

3. Per le figure bisogna estrarle dal documento, se non si dispone di file appositi; nel caso della composizione di un testo "nuovo", tutta questa operazione potrebbe essere dovuta all'autore che non conosce LaTeX e che consegna al compositore il testo scritto, per esempio, con Word e, pur avendo inserito le figure nel file `.doc` ha consegnato al compositore anche le foto e le altre illustrazioni mediante file appositamente allegati. Se non fosse così il compositore risparmierà molto tempo se richiede questo materiale all'autore, ricordando che per il materiale a matrici di pixel, come le fotografie, la densità di punti al pollice deve essere almeno di 300 dpi.

   Se invece non si dispone delle illustrazioni originali, estrarle dal documento vuol dire perdere enormemente in dettaglio e si è fortunati se si riesce ad estrarre immagini a 100 dpi; la qualità scende vistosamente approssimativamente in proporzione al quadrato del rapporto delle densità; scendere da 300 dpi a 100 dpi vuol dire perdere in qualità di un fattore di circa 10 volte!

4. Per le tabelle si può procedere sia ricomponendo ogni tabella cella per cella, reimpostandola in modo più professionale; oppure la si può copiare integralmente e, dopo averla incollata, se il proprio *shell editor* lo consente, si possono eseguire le sostituzioni di tutti i caratteri `<TAB>` (ASCII 09) con il carattere `&` ricorrendo alle *regular expressions*, un modo avanzato di usare gli shell editor che consente di fare delle correzioni selettive anche molto elaborate. Bisogna poi marcare i fine riga della tabella con `\\` e, eventualmente, inserire qualche filetto.

5. Per le formule nel testo si è già detto; per quelle in display o le si ricompone a mano partendo da zero, oppure può succedere che l'*equation editor* usato per comporre il file da trasformare abbia salvato oltre all'immagine della formula, anche il codice usato per comporla; allora conviene copiare questo codice che spesso e volentieri è codice TeX, o gli assomiglia molto. La collezione di programmi da ufficio OpenOffice.org dispone di un *plug in* che mette a disposizione dell'utente un *equation editor* basato su LaTeX; il codice LaTeX, quindi, può essere copiato e incollato direttamente. Se si ha la competenza per farlo, si possono anche correggere gli errori delle formule

da trasferire sul documento LaTeX, sempre prendendone nota e informando l'autore per chiedere conferma della correttezza della correzione. Se non c'è autore a cui chiedere, al massimo si può ripetere la formula originale senza correzione, inserendo la formula corretta in nota accompagnata da una frase adeguata.

6. Le note vanno copiate e incollate durante la lavorazione dei singoli capoversi; è molto più difficile inserire le note alla fine dopo aver sistemato l'intero testo; non è complicato, perché basta incollare il testo della nota fra le due graffe che delimitano l'argomento del comando `\footnote`.

   È molto più delicato convertire le note di tipo umanistico per i riferimenti bibliografici; come è noto gli umanisti sono soliti annotare i riferimenti bibliografici in calce alla pagina usando il meccanismo delle note; questo meccanismo, per conservare spazio, usa spesso abbreviazioni di tipo *Ibidem* oppure *loc. cit.* per riferirsi all'opera citata immediatamente prima oppure non sequenzialmente ma nella stessa pagina. Queste note vanno trattate con comandi `\cite` ben congegnati e in accordo con gli appositi stili bibliografici per la bibliografia. Di conseguenza questo genere di note va convertito avendo bene chiaro in mente il tipo di stile bibliografico e il tipo di citazione, in modo da usare il pacchetto adatto ad affrontare questo tipo di composizione.

7. Per i caratteri non ASCII si può procedere capoverso per capoverso, oppure alla fine dell'introduzione dell'intero testo da convertire; qui le esigenze sono opposte a quelle suggerite sopra per procedere un capoverso alla volta. Infatti i caratteri da convertire sono sorprendentemente numerosi, specialmente quando si converte un file originariamente scritto in formato `.doc`.

   Naturalmente è possibile affrontare il problema con un opportuno encoding dei caratteri in input e un altrettanto opportuno encoding del font con cui comporre. Probabilmente questo è un caso in cui usando sia per il file sorgente sia per il font da usare in uscita l'encoding `utf-8`[2] si risparmia molto lavoro. Chi scrive ha già convertito un numero non indifferente di documenti lavorando "a mano", ma ha sempre evitato di usare la codifica `utf-8`. I motivi sono molteplici, alcuni giustificati, altri difficili da spiegare, nel senso che si potrebbero riassumere nell'antico proverbio: "Chi lascia la via vecchia per la nuova, sa quel che lascia e non sa quel che trova". Non molto intelligente, ma probabilmente nessuno è indenne da questo tipo di ragionamenti.

   Tra le altre cose bisogna disporre di uno shell editor capace di gestire la codifica `utf-8`; per molti anni chi scrive ha usato WinEdt che non era

[2] La codifica in input per usare l'encoding `utf-8` è ottenuta specificando l'opzione *utf8*; per l'uscita sarebbe preferibile usare font codificati in UNICODE; per questo è molto meglio usare il programma XeTeX, oppure luatex, che gestiscono direttamente i font contenenti più di 256 caratteri; le versioni come XeLaTeX e lualatex, che contengono la parte La, sono generalmente compatibili con il mark-up di LaTeX.

in grado di gestire questa codifica. Oggi gli shell editor capaci di gestire questa codifica sono numerosi; tuttavia anche con questi editor non è banale introdurre i segni non presenti in tastiera; è vero, con questa codifica si può scrivere anche in cinese, ma bisogna disporre di una adeguata interfaccia per scegliere gli ideogrammi da inserire nel testo in modo che non sia necessario ricercare l'ideogramma voluto in una enorme tabella che elenca le decine di migliaia di ideogrammi disponibili.

Questo problema è molto ridotto con gli alfabeti latini, tuttavia anche con questi alfabeti i segni non letterali o paralfabetici che potrebbe essere necessario gestire sono molto numerosi. Un breve elenco non guasta:

- l'apostrofo spesso è un carattere speciale, leggermente inclinato, non verticale come appare sullo schermo quando si usano i caratteri ASCII;
- le virgolette alte sono spesso virgolette simmetriche (`"`), invece che asimmetriche (“ e ”); talvolta sono correttamente asimmetriche, ma sono rappresentate dagli appositi caratteri non ASCII, che generalmente non sono accettabili direttamente con un encoding come quello suggerito in tutto questo testo (`latin1`); esse vanno quindi convertite nelle apposite sequenze delle legature `‘‘` e `’’`, oppure sostituite con macro che svolgano correttamente il loro lavoro dietro le quinte;
- non è il caso di preoccuparsi per gli apici delle note; questi vengono direttamente sostituiti e rigenerati quando viene usato il comando `\footnote`;
- sono invece da gestire adeguatamente gli apici di nota eseguiti mediante caratteri paralfabetici, come l'asterisco, la spada, la doppia spada, eccetera; in questi casi conviene ricorrere ad appositi pacchetti che consentono di gestire agevolmente la composizione delle note; la documentazione fornita con il sistema TEX offre diversi pacchetti e ognuno sceglierà quello che risulta maggiormente adatto alle proprie esigenze;
- talvolta è presente il segno dei gradi, °; più raramente le frazioni a barra del tipo $1/2$, $3/4$, e simili. Il segno dei gradi potrebbe risultare compatibile con l'encoding `latin1`, ma si ritiene che per “pulizia” sia meglio renderlo correttamente con la sua rappresentazione mediante i segni disponibili con LATEX, che dispone di `\textdegree` per la composizione testuale e `^\circ` per la matematica; si è già esposto nei capitoli precedenti come definire nuovi comandi che consentano di automatizzare la scelta fra modo testo e modo matematico; se si parla di gradi Celsius[3], °C, bisogna correttamente inserire lo spazio fine,

[3] Qui si è barato un pochino; nel file che contiene le macro specifiche per la composizione di questo testo, si sono usati comandi diversi per la composizione testuale dei gradi Celsius, `\textcelsius`, ℃, e `$^\circ\mathrm{C}$`, °C, per la matematica. Si osserva che i due segni sono leggermente diversi; negli esempi qui riportati si è sempre usato il segno matematico reso il più possibile simile a quanto si ottiene con il segno testuale.

che invece non bisogna inserire quando si parla di gradi "angolari"; si confronti l'angolo di 30° con la temperatura di 30 °C;

- gli apici ª e º degli ordinali femminili e maschili vanno sostituiti anche questi con gli appositi comandi disponibili con l'opzione *italian* di `babel`: `\ap{a}` e `\ap{o}`;
- i deponenti (e i rari ascendenti) da comporre con un carattere non inclinato in modo matematico vanno comunque corretti tutti quanti; è difficile che chi ha scritto un documento con Word o con Writer abbia avuto l'attenzione di distinguere con il font giusto i deponenti attributivi dai pedici letterali "variabili"; questa correzione, comunque necessaria, richiede la comprensione della matematica da parte del compositore;
- Certamente si possono presentare altri caratteri non ASCII nei file sorgente, ma quelli più frequenti sono quelli illustrati sopra; non bisogna preoccuparsi delle lettere accentate, che a rigore non sono ASCII, perché solitamente esse non appaiono come incompatibili nella finestra di editing; se lo fossero, le parole che le contengono risulterebbero marcate come ortograficamente errate, quindi vi si può intervenire caso per caso.

A chi scrive non è mai capitato di dover intervenire per le vocali accentate della lingua italiana (salvo l'eventuale correzione degli accenti gravi o acuti per la lettera 'e' dove precise norme prescrivono l'uno o l'altro accento), ma potrebbe verificarsi in lingue diverse dall'italiano, oppure, più subdolamente, in parole straniere inserite in un contesto italiano, perché in questi casi il correttore le dà comunque ortograficamente errate (di solito i correttori ortografici sono impostati per una sola lingua alla volta) e non è immediatamente chiaro se la cosa sia solo dovuta ad una parola non appartenente alla lingua di default, oppure se sia dovuta alla presenza di una lettera con diacritici che non corrisponde all'encoding preimpostato; in questi rari casi si può procedere senza preoccuparsi dell'esito della composizione; durante la compilazione eseguita con pdflatex se il segno non ASCII non è riconosciuto dalle definizioni corrispondenti all'opzione data al pacchetto `inputenc`, viene segnalato un *warning* in corrispondenza del carattere non riconosciuto ed esso viene sostituito con un punto interrogativo; di conseguenza prima di eseguire nuovamente pdflatex basta sostituire la lettera incriminata con la sua definizione esplicita mediante gli appositi comandi per gli accenti e i diacritici in particolare.

Dovendo citare la locuzione irlandese[4] *Rí Tea̋mrać* bisogna scrivere esplicitamente `Rí Tea\.mra\.c`, perché altrimenti i caratteri puntati vengono

[4]Questa locuzione è stata ripresa dal testo di Peter Flynn [22]. In quel testo Flynn dice anche che non esiste una chiave di tastiera che consenta di inserire una 'i' priva di puntino e che perciò bisogna usare la sequenza `\i`; questo è vero per la lingua celtica usata in Irlanda e per la lingua turca; per le 'i' accentate questo era vero fino ad una decina di anni fa, quando per tutti i diversi encoding è stato dichiarato che la macro dell'accento seguita da una 'i' "normale" (col

sostituiti con punti interrogativi (*Tea?ra?*) in quanto incompatibili con l'encoding `latin1` con cui è stato composto questo testo.

Per la conversione di questi caratteri, indipendentemente dalla capacità dell'editor di gestire caratteri diversi da quelli corrispondenti al suo encoding impostato, basta copiare una qualunque istanza del carattere da sostituire, inserirla nel campo di ricerca della finestra "cerca e sostituisci", e inserire il testo o il carattere o la macro sostitutiva nel campo di sostituzione per poi eseguire una sostituzione globale. Di solito questa operazione, da ripetere per ogni carattere non ASCII, è risolutiva, anche se non guasta controllare una seconda volta che cosa sia stato effettivamente sostituito e dove.

Nel fare queste operazioni di sostituzione e di correzione si inseriscono anche nei successivi capoversi o nelle successive sezioni del testo ricomposto tutte le informazioni di mark-up necessarie, le etichette per le informazioni incrociate, i comandi per la generazione degli indici e delle varie liste di oggetti flottanti; bisogna dunque aggiungere con la sintassi di LaTeX quelle informazioni che non potevano essere contenute dentro i file di puro testo o formattati secondo il formato `.doc`.

L'esecuzione del programma pdflatex non dà più problemi di quelli che si hanno nella prima esecuzione di un qualunque file sorgente; certamente compariranno delle cose non errate sintatticamente, ma che non corrispondono esattamente al quel che si desiderava comporre; certamente bisognerà agire sugli oggetti flottanti per trovare loro una posizione conveniente in relazione ai parametri di collocamento automatico di cui si è parlato negli appositi capitoli. Ma questi problemini sono quelli che si incontrano sempre con qualunque documento complesso.

Per evitare di dover ricercare grossi errori, come per esempio l'assenza di una graffa chiusa o della fine di un lungo ambiente di qualunque tipo, conviene compilare il documento ogni pochi capoversi, non necessariamente ad ogni capoverso aggiunto al file in lavorazione; ecco perché si è consigliato di convertire un capoverso alla volta per apportare le modifiche di poco conto, mentre la compilazione ad ogni gruppo di capoversi, due o tre pagine alla volta, permette di scovare le imperfezioni di più largo respiro e di eseguire le conversioni dei caratteri non ASCII meno frequentemente che non un capoverso alla volta.

Chi scrive ha convertito un certo numero di libri in questa maniera; un paio di questi libri erano atti di congressi o di simili conferenze di argomento tecnico scientifico. Specialmente per questi ultimi, i cui contributi erano stati scritti da ogni presentatore di memorie, ma con stili tutti diversi, con abilità più o meno grandi nell'uso di Word o di Writer, con competenze di vario livello in merito alle immagini allegate ai loro lavori; con formule composte spesso in violazione di ogni possibile norma di scrittura della matematica tecnico scientifica e senza servirsi di *equation editor* validi, hanno presentato notevoli problemi; sicuramente

puntino) produce direttamente e correttamente la 'i' accentata con l'accento che sostituisce il puntino. Forse non è vero con tutti i possibili accenti, ma certamente lo è per i più comuni.

non sarebbe stato possibile eseguire nessuna conversione in forma automatica o semiautomatica.

23.2 Conversione automatica

23.2.1 Documenti in formato `.doc`

Chi scrive ha provato ad usare programmi come rtf2latex2e; qualunque word processor, come Word o Writer può aprire un documento in formato `.doc` e salvarlo in formato `.rtf` (Rich Text Format). Il programma citato, rtf2latex2e, scaricabile dalla rete per i tre sistemi principali, sembra che consenta di trasformare qualunque documento RTF, gestendo anche le figure e trattando le eventuali formule in esso contenute come immagini. L'esperienza individuale può essere molto varia con questo programma e chi scrive non è riuscito ad ottenere granché, probabilmente per sua inesperienza.

Esiste anche un plug-in per Microsoft Word che si chiama word2tex, usabile solo su piattaforme Windows; dagli archivi ufficiali CTAN è scaricabile un programma eseguibile con lo stesso nome ma datato 1988, quindi vecchiotto, che dovrebbe fare le stesse cose. In rete c'è questo plug-in commerciale aggiornato al 2008, nella forma di uno shareware, la cui licenza singola, da pagare dopo i 30 giorni di prova, non è proprio a buon mercato, ma sembra che ne valga il costo, specialmente se si esegue la conversione Word $\longrightarrow$ LaTeX con una certa frequenza. Le prove che chi scrive ha visto già eseguite da altri[5] sembrano eccellenti; il codice LaTeX è corretto e senza sbavature, anche se in realtà si deve eseguire qualche piccolissimo intervento a mano per dare qualche piccolo ritocco; il file convertito, però, è subito compilabile e il documento prodotto, a parte i font e qualche altro piccolissimo dettaglio, è sostanzialmente identico all'originale.

Il plug-in word2tex si presenta nei menù di word come una opzione del comando `Salva con nome` dove fra gli altri formati compare anche il formato TeX (in realtà vuol dire LaTeX); se il file sorgente contiene figure o equazioni, queste vengono correttamente convertite in modo compatibile se sono presenti sulla macchina Windows alcune librerie dinamiche sufficientemente recenti; tuttavia per questi dettagli è meglio leggere la documentazione sul sito ufficiale `http://www.chikrii.com/`.

Invece i programmi per Linux AbiWord e Kword possono aprire un documento `.doc` e salvarlo ("Salva con nome") in formato `.tex` con il mark-up di LaTeX. È chiaro che questi programmi, essendo nativi di Linux e, specialmente il secondo, essendo legati al desktop KDE, funzionano solo su macchine Linux[6]; in ogni modo nessuno vieta di installare su una macchina Windows il programma CygWin, che serve per simulare un ambiente UNIX su una macchina Windows, e

[5] Chi scrive lavora su un Mac; ma il plug-in descritto qui non è installabile nella versione di Microsoft Office per Mac.

[6] Sembra che AbiWord offra anche una versione adatta alle piattaforme Windows; per maggiori informazioni si veda nel sito Internet di AbiWord.

successivamente installare dentro questo ambiente di simulazione uno dei due programmi in questione. Sulle macchine Mac con il sistema operativo OS X, non ci sono particolari problemi; o si scarica l'apposito programma di installazione per Mac, oppure si scarica il file sorgente e si genera l'applicazione compilando il file sorgente; questo è uno dei vantaggi conseguenti all'adozione del sistema UNIX da parte della Apple. Quindi in sostanza i due programmi nominati sono usabili su tutti e tre i sistemi operativi maggiori.

Tuttavia... alla data odierna (marzo 2010) chi scrive ha provato ad usare entrambi i programmi summenzionati su una macchina Linux dotata di sistema operativo Ubuntu 9.10.

Con AbiWord si è aperto un primo file `.doc` che il programma ha importato direttamente senza problemi; lo si è salvato in formato LaTeX; si è aperto questo nuovo file, ma la conversione non ha dato luogo ad una compilazione corretta, e l'intervento a mano per renderlo compilabile è stato decisamente pesante. Si è aperto un altro file dello stesso formato, questa volta un poco più complesso, contenente anche l'indice generale. Di nuovo l'editing per renderlo compilabile con pdflatex ha richiesto non poco lavoro. Il motivo principale è che il file `.doc` originale era stato composto su una macchina Windows, il cui fine riga è caratterizzato da due byte; uno indica di andare a capo e l'altro di iniziare una nuova riga. Nelle macchine UNIX i file di testo hanno il fine riga marcato da un solo byte, quello di inizio di una nuova riga. Benché AbiWord mostri correttamente sullo schermo il contenuto del file `.doc`, quando lo converte in formato LaTeX si comporta come se non riconoscesse entrambi i byte di fine riga del formato "Windows", ma ne riconosce uno solo, e racchiudesse l'altro in un ambiente *flushleft*; questo succede specialmente nelle parti preliminari del documento che hanno diversi spazi verticali fra capoversi introduttivi. Certo, sapendolo, si sa che bisogna cancellare un grande numero di questi ambienti contenenti solo una riga vuota, ma richiede del tempo e molta attenzione per essere sicuri di non cancellare "troppo". Bisogna in ogni caso modificare le impostazioni di default del preambolo e molte righe di mark-up interne. Fattibile, ma faticoso.

Con Kword le cose sono simili, ma leggermente migliori; il dialogo che il programma apre per svolgere la conversione richiede diverse indicazioni che consentono anche di scegliere l'encoding del file sorgente. La scelta della lingua principale del documento non sembra avere effetto, mentre con AbiWord non c'è personalizzazione e la lingua di default è comunque l'inglese. Entrambi specificano il formato americano *letter* per la carta, ma è immediato modificarlo in A4.

Insomma: entrambi i programmi eseguono la conversione, ma entrambi producono in uscita dei file LaTeX che richiedono non poco lavoro per renderli compilabili.

Chi scrive ha usato anche altri programmi di cui si parla nelle FAQ britanniche, ma i risultati non sono stati per niente soddisfacenti.

Non bisogna stupirsene più di tanto; i file `.doc` non contengono nessuna informazione in merito alla struttura del documento, ma contengono informazioni in merito solamente all'aspetto visuale del testo; quindi anche se questi programmi convertono il file dal formato `.doc` al formato `.tex` con il mark-up di LaTeX, i file

così prodotti richiedono non poco lavoro per produrre dei documenti composti tipograficamente in modo corretto.

Per la componente Writer di OpenOffice.org e LibreOffice esiste l'estensione Writer2LaTeX che non è installata di default, ma ogni utente di OpenOffice.org e di LibreOffice sa come reperirla dal sito ufficiale di quelle suite; scaricare questa estensione e installarla è una questione di pochi click; una volta installata l'estensione, la voce "Export" del menù "File", consente di esportare il file in formato LaTeX; una finestra di dialogo viene aperta per specificare le impostazioni di default.

Il file da convertire (chi scrive ha provato a convertire un file con indirizzi internet, liste e sezionamenti) viene convertito perfettamente e pdflatex lo compila senza errori conservando abbastanza bene l'aspetto che il testo aveva nel formato `.doc`; la lingua viene riconosciuta correttamente così come i caratteri speciali, accettati come caratteri non ASCII che vengono riconosciuti nell'encoding *latin1*. Se c'è qualche modesta critica da fare è quella che viene caricato un gran numero di pacchetti esterni, molti dei quali servono solo per riconoscere caratteri che la maggior parte delle volte non sono stati effettivamente usati nel file sorgente. Inoltre l'apostrofo viene sempre convertito in una virgoletta semplice alta di chiusura, ma tutto sommato non è difficile sostituire la stringa `{\textquoteright}` in un semplice apostrofo quale quello che si ottiene dalla tastiera; questa sostituzione, però, è solo estetica, perché la sua omissione non produce danni di nessun genere, salvo rendere leggermente più faticosa la lettura del file sorgente[7]. Generalmente bisogna anche cambiare il formato della carta da `letterpaper` a `a4paper`. Per il resto questo convertitore/esportatore dal formato `.doc` al formato `.tex` sembra eccellente.

Merita aggiungere che gli sviluppatori di Writer2LaTeX stanno lavorando su una ulteriore estensione, Writer4LaTeX, che dovrebbe essere già inclusa na partire dalla versione 1.2 di Writer2LaTeX, e che dovrebbe integrare completamente Writer con LaTeX (latex e pdflatex) in modo da svolgere le funzioni di *shell editor* per LaTeX; insomma in un certo senso svolge le funzioni di LyX, con un suo formato di default e con la possibilità di compilare direttamente con latex o pdflatex e di visualizzare direttamente il frutto della composizione senza uscire da Writer. Il messaggio che sul sito `http://writer2latex.sourceforge.net/index4.html` accompagna le altre informazioni su questa nuova estensione richiama ancora una volta che sia Writer sia LyX hanno delle capacità limitate (*a finite feature set*), ma che l'utente esperto può sia con LyX sia con Writer inserire codice in linguaggio TeX/LaTeX in modo da sopperire alle limitazioni del programma usato per la predisposizione del file sorgente.

Il sito Wiki di LyX, `http://wiki.lyx.org/Tools"/Word2LyXMacro` contiene alcune macro Word che permettono a Word (2007 e 2000) di "autoconvertire" un

[7] Chi scrive si è rivolto al creatore di questa estensione per segnalargli questo inconveniente; egli ha prodotto una versione "sperimentale", che lo scrivente ha ha avuto a disposizione prima della distribuzione ufficiale, e questo inconveniente relativo all'apostrofo non si presenta più; probabilmente quando il lettore leggerà questa nota, in rete sarà disponibile la versione ufficiale modificata.

documento `.doc` in un corrispondente documento `.lyx`; questo può venire aperto con LyX ed essere salvato in formato `.tex` con il mark-up di LaTeX; chi scrive ha provato a montare il file con le definizioni delle macro Word sulla versione Mac di questo programma (Microsoft Office 2004), ma l'esperimento è fallito; benché le macro siano state salvate e correttamente individuate dal programma, esse non hanno dato luogo che a un errore grave. Tuttavia l'insuccesso è dovuto ad una incompatibilità del set-up della suite Microsoft Office per Mac con il corrispondente set-up su una macchina Windows. Infatti su questa macchina la stessa procedura di installazione (Word XP) dà luogo a macro eseguibili che convertono correttamente il file `.doc` in un file `.lyx`. Aprendo il file con LyX e salvandolo in formato `.tex` con il mark-up di LaTeX, si osserva una conversione quasi perfetta subito compilabile senza errori significativi.

Gli "errori" sono in realtà impostazioni non desiderate nel preambolo. Per esempio il pacchetto *hyperref* è caricato con l'opzione *dvipdfm*, mentre sarebbe meglio che questa opzione sia presente solo nel file da compilare con latex, non con pdflatex: basta cancellare l'opzione. Curiosamente viene impostato l'encoding del file sorgente con l'opzione *latin9*[8], ma non è un problema, se lo si desidera, cambiare l'opzione in *latin1*. Ancora: viene usata di default una classe della collezione KOMA-script, ma viene specificata come opzione globale di default la lingua inglese. Come si vede piccole cose che possono venire modificate in modo semplicissimo.

Si tenga presente che la macro Word che usa questo procedimento non converte le equazioni; queste vanno quindi reintegrate a mano. Non solo ma le figure eventualmente presenti nel file `.doc` non devono essere effettivamente incluse, ma devono essere collegate con un link, cioè vengono rese visibili nel documento originale mediante un link alla loro posizione nella cartella dove risiede il file `.doc` attraverso il comando di menù LinkToFile.

23.2.2 Documenti in formato PDF

Bisogna ancora ricordare che AbiWord e Kword possono aprire o importare documenti PDF; una volta aperti, questi documenti possono venire salvati in formato `.tex` adatti al mark-up di LaTeX. Va da sé che dai documenti PDF non si riescono a convertire adeguatamente le formule; quindi queste vanno ricomposte oppure trattate come detto qui di seguito

Uno dei vantaggi di usare un file PDF da cui partire, specialmente se l'autore ha incluso le figure partendo da file adatti, con una definizione sufficientemente elevata, è che queste figure talvolta possono venire estratte, ma che comunque possono venire ritagliate dalla pagina PDF dove compaiono; lo stesso si può fare per le formule se queste sono state composte correttamente e con i font "giusti", cioè che non stonino con i font del documento.

[8] L'opzione *latin9* dovrebbe consentire l'uso immediato del simbolo dell'euro, senza bisogno di ricorrere al pacchetto *textcomp*; non è però chiaro in che ordine debbano essere specificate le chiamate ai vari pacchetti, per cui si possono incontrare errori oppure l'euro viene sostituito da un punto interrogativo.

Può darsi che l'operazione si possa fare nativamente anche con altri programmi, ma sui Mac OS X il programma Preview (Anteprima se il sistema operativo è in italiano) permette di selezionare una parte di pagina, per esempio una figura o una formula, di copiarla nel "clipboard" e poi, agendo sulla voce del menù File: "New from clipboard", incollare quanto si era copiato in un nuovo file che può venire salvato in qualunque formato fra i tanti di cui il programma è dotato, in particolare in formato PDF, dando al file un nome adeguato; questo può poi venire incluso nel documento in lavorazione con il solito comando `\includegraphics`, già descritto nei capitoli precedenti. Il programma Preview consente di scontornare l'immagine prima di salvare il file, per cui, una volta terminata l'operazione, quanto deve venire incluso è già pronto senza bisogno di ricorrere alle operazioni di "cropping" già descritte nei capitoli precedenti.

Ovviamente questa operazione di taglia e incolla delle figure e, talvolta, delle formule, può essere eseguita anche con disegni o altre forme di illustrazioni prodotte da altri programmi. Chi scrive solitamente usa questo metodo per estrarre i diagrammi di vario genere da file prodotti con excel; il risultato è solitamente molto buono. Anche se questa non è una operazione per convertire il formato di un documento, è pur sempre un tipo di operazione che occorre fare spesso quando si compongono documenti il cui materiale ha provenienze diverse.

Chi scrive non è al corrente di tutti i possibili programmi per aprire ed elaborare file in formato PDF con le prestazioni di Preview appena descritte; egli è al corrente del programma Skim sempre per il sistema operativo Mac OS X, che svolge le stesse funzioni di Preview ed alcune altre, molto comode, fra le quali la possibilità di essere 'sincronizzato' con lo *shell editor*. Ne esistono di sicuro anche per gli altri sistemi operativi, ma è bene che ognuno se li cerchi e se li installi sapendo bene che cosa vuole ottenere.

Una informazione interessante, però, è che dalla fine del 2009 la Sun Microsystems ha messo a disposizione per il suo StarOffice e per OpenOffice.org un programma di estensione chiamato pdfimport, da installare come tutti gli altri plug-in di estensione per quei sistemi. Con questa estensione installata sarebbe possibile importare file PDF nella sua versione Draw, la parte di OpenOffice.org destinata ai disegni, e i comandi di editing disponibili sono tali da poter introdurre semplici correzioni (visto che senza un programma dedicato è molto difficile – parole della Sun – apportare modifiche ai file in formato PDF). Tuttavia, una volta importato il file e introdotto l'intero documento in un file del Writer di OpenOffice.org, non dovrebbe essere molto complicato esportare il file in formato LaTeX compatibile, se è già stata installata l'estensione Writer2LaTeX.

Chi scrive non ha materialmente provato questa strada, sia perché a tutt'oggi (luglio 2010) il programma è distribuito come versione 1.0, ma forse sarebbe meglio considerarlo una versione beta avanzata; sia perché apparentemente, se la cosa funziona, ogni riga viene salvata come una scatola o riga a sé stante, quindi bisognerebbe eliminare moltissime dichiarazioni di ambiente. Tuttavia, anche se l'oprazione sembra richiedere un grosso lavoro, vale la pena di tenerne conto perché in alcune circostanze potrebbe essere meglio che eseguire la conversione a mano.

23.2.3 Documenti in formato XML

Da diversi anni Microsoft Word può salvare i suoi documenti in una sua versione di formato XML (eXtended Markup Language); l'estensione dei documenti in questo formato è `.docx`; dal 2003 il programma può salvare anche in formato `.xml`, ma bisogna chiederlo esplicitamente al momento di salvare il documento.

OpenOffice.org Writer ha come formato standard l'XML e i suoi file salvati in questo formato hanno l'estensione `.odt` se si tratta di testo con poca o nulla formattazione, oppure in formato `.sxw` se si tratta di un documento strutturato. Siccome Writer può salvare anche nei formati della suite Microsoft Office, Writer può salvare anche nei formati `.docx` e `.xml`.

Tra i programmi già elencati bisogna ricordare che AbiWord può nativamente aprire i file XML prodotti con qualunque estensione da Microsoft Word e da OpenOffice.org Writer; quindi può di fatto aprire qualunque versione standard o meno standard di documento XML, e li può salvare in formato `.tex` con il mark-up LaTeX,

Anche Kword può aprire file con i formati XML di base o le variazioni di Microsoft Word e di OpenOffice.org Writer. Anche Kword può salvare i suoi documenti in formato `.tex` con il mark-up di LaTeX.

Il programma LyX, invece, non apre questi formati XML in modo diretto; è un peccato, perché il codice LaTeX prodotto da LyX è decisamente il migliore fra quelli ottenuti per conversione automatica.

Però, almeno sul Mac, è disponibile, o scaricabile dal sito di Microsoft, l'applicativo Open XML converter che consente di convertire i file XML in formati validi per versioni precedenti a quella del 2003 della suite Microsoft Office; tra questi, ovviamente, c'è il tradizionale formato `.doc` con il quale si può operare come descritto nei paragrafi precedenti.

I file in formato `.docx` sul Mac possono venire aperti anche con la semplice applicazione TextEdit che può salvare il file in formato `.rtf`, sul quale si può operare come descritto nei paragrafi precedenti.

Anche il potentissimo editor emacs può convertire i file XML in file adatti a LaTeX; sembra anzi che esso abbia funzionalità particolarmente avanzate; chi scrive però non ne ha esperienza diretta, ma si rifà solo a quanto Peter Flynn afferma nel suo testo [22].

Peter Flynn indica anche un'altra strada per convertire i file in formato XML nel formato adatto a LaTeX; la procedura descritta da Flynn nel suo paragrafo 10.1.1 "Getting LaTeX out of XML" è abbastanza lunga da descrivere, anche perché Flynn scrive delle macro in XLM e le usa per convertire direttamente in formato adatto a LaTeX, usando un programma chiamato Saxon, un programma Java per l'elaborazione di documenti XSLT. Secondo Flynn, questo sarebbe il metodo migliore per eseguire le conversioni, ma richiede delle capacità d'uso dei PC che chi scrive ritiene molto superiori alla media degli utenti, comunque molto superiori alle sue capacità.

Merita ricordare il programma dblatex; si tratta di un comando senza interfaccia grafica da lanciare da linea di comando; esso apre i file in forma-

to XML e li trasforma direttamente in file LaTeX. Esso gestisce convenientemente gli *style sheets* necessari per la conversione. Esso può venire scaricato dal sito `dblatex.sourceforge.net` e le istruzioni di lavoro sono in `http://dblatex.sourceforge.net/doc/`. Esso è fornito di comandi di pre- e post-processing che consentono di eseguire tutte le operazioni per comporre il documento o direttamente attraverso pdflatex o indirettamente passando attraverso il formato PS. Durante queste operazioni le figure, per esempio, vengono convertite nei formati richiesti da pdflatex o da latex (o meglio, da dvips). In questo modo chi converte il file XML si ritrova alla fine sia con il documento composto, sia con un sorgente LaTeX che può tranquillamente modificare per ottenere un tipo di composizione diversa. Probabilmente questo pacchetto esegue le stesse operazioni descritte da Flynn e menzionate molto brevemente poco sopra, ma le esegue in modo automatico, senza o con poco intervento da parte del compositore. È interessante notare che l'autore di dblatex ha predisposto anche un programma "gemello", dbcontext, per convertire i file XML in modo da poterli comporre con il programma context.

23.3 Documenti in collaborazione

Merita riprendere il discorso sui documenti composti da più autori. Per questo scopo sarebbe bene usare strumenti opportuni, piuttosto che fare artigianalmente tutto quanto. In ogni caso è necessario che ci sia uno dei coautori che si comporti da capogruppo e che si occupi dell'omogeneizzazione dei contributi di tutti gli altri autori; lo stile letterario e lo stile tipografico dei vari contributi dovrebbe essere identico; si può forse omogeneizzare lo stile tipografico, è più difficile farlo con lo stile letterario.

Ovviamente prima di cominciare un lavoro fra più autori è bene che il coordinatore abbia cura di concordare con gli altri autori un piano di lavoro; una scaletta o un indice provvisorio a cui i coautori dovrebbero adeguarsi affinché il lavoro sia ordinato, privo di ripetizioni e omogeneo. Certamente un gruppo di autori che si accinga a creare un lavoro a più mani sa come scegliersi il coordinatore e come organizzarsi, quindi qui non si insiste più di tanto su questa fase preliminare.

Tuttavia il coordinatore deve conoscere bene i software a disposizione e deve saperli usare con competenza per ottenere un lavoro finale di qualità.

Per raggiungere questo obiettivo ci sono degli strumenti utili che possono essere usati anche per comporre un documento scritto da un solo autore, ma qui probabilmente è meglio sottolineare l'utilità nel lavoro collaborativo.

Oggi l'uso del computer semplifica il flusso di lavoro, nel senso che è facile presumere che i singoli contributi siano prodotti fin dall'inizio in formato digitale, ma ciò non significa che una volta raccolti essi siano già *pronti e a posto* per la successiva fase di composizione del documento finale. È difficile che questo accada. È più consueto, invece, che si presenti la necessità di comporre una prima bozza che servirà per valutare l'opera nella sua interezza e procedere successivamente

con il completamento e l'armonizzazione di tutti i capitoli. In parte ciò è dovuto al fatto che i differenti autori lavorano, perlomeno inizialmente, ognuno per conto proprio, senza vedere cosa e come fanno gli altri, ed in parte perché un elaborato si sviluppa nel tempo, in condizioni e stati d'animo differenti e mutevoli.

Per limitare le problematiche sopra accennate il coordinatore ed i vari coautori possono avvalersi di strumenti informatici e di metodi in grado di ottimizzare e semplificare il flusso di lavoro.

Nei successivi paragrafi verranno analizzati alcuni programmi scelti principalmente in base all'esperienza di chi scrive[9]. Conseguentemente non si tratta del sistema o dei programmi migliori in senso assoluto ma di sistemi ed applicazioni adoperati e vagliati nel tempo dall'autore, spesso in conseguenza di fatiche, di difficoltà, di errori commessi, di cui si è fatto tesoro in relazione:

- alla loro versatilità per un uso collaborativo;
- alla loro capacità di gestione di testi articolati;
- all'integrazione con LaTeX;
- alla loro integrazione con sintassi (*markup*) semplici quali *multimarkdown* (MMD) e *markdown* (MD) a cui far ricorso per produrre contributi testuali da utilizzare in forme diverse: per la stampa (su pdf e su carta), per la pubblicazione sul web (html), per la preparazione di presentazioni.

La compresenza di più autori comporta almeno un paio di valutazioni preliminari, fondamentali per delineare i successivi percorsi operativi.

1. Lavorano tutti con la stessa piattaforma informatica oppure operano su piattaforme differenti?
2. Qual è il livello di competenza informatica di ciascun coautore?

È importante, ad esempio, sapere fin dall'inizio del lavoro, se tutti i collaboratori conoscono LaTeX, oppure MMD e/o MD, per definire a priori in che modo verranno scritti e consegnati i file di ciascuno e che tipo di applicativi si potranno adottare per la composizione della bozza. Le differenze in termini di tempo e di efficienza possono essere significative per valutare, a priori, l'entità dell'impegno che il coordinatore dovrà affrontare e per stimare i conseguenti tempi di elaborazione del lavoro.

Nei successivi sotto-paragrafi segnaliamo alcuni software utili a partire dalla fase iniziale di produzione delle bozze preparatorie fino alla produzione del documento finito, compilato in LaTeX. Le scelte indicate, incentrate principalmente sul sistema operativo Mac OS X, non vanno interpretate come un'intrinseca indicazione di preferenza di una piattaforma informatica rispetto ad altre; vanno invece intese come una limitazione dell'autore di questo contributo che non conosce a sufficienza altri sistemi per poterne scrivere con eguale competenza.

[9] Questa sezione è stata gentilmente messa a disposizione da Federico Morchio.

23.3.1 GIT

Il sistema GIT si basa sul fatto di comporre i propri file dentro un ambiente di sviluppo che tiene traccia delle versioni e delle modifiche fatte; allo sviluppo di un unico file o di molti file correlati possono partecipare diverse persone che contribuiscono con le loro versioni o le loro modifiche, ma c'è una sola persona che può accettare le modifiche proposte ed integrarle nel set di file, conservando però le versioni precedenti, cosicché si possano ripristinare selettivamente parti di file precedenti o interi file.

Come si può facilmente capire, questa è esattamente la soluzione alla situazione descritta nel paragrafo precedente, ma richiede che tutti i coautori conoscano LaTeX con lo stesso livello di competenza.

Non c'è ragione di entrare qui nello specifico di questa soluzione poiché nella sezione `http://www.guitex.org/home/it/guide-tematiche` del sito GUIT è scaricabile un'ottima guida dedicata, *GIT 4 LaTeX* che spiega con chiarezza come mettere al sicuro, anche su server remoti, i file in collaborazione. Perciò si rinvia direttamente a quella guida.

23.3.2 Organizzazione minimale

Se si vuole evitare di ricorrere a software specialistici o proprietari un'organizzazione pratica e logica del lavoro di gruppo si ottiene utilizzando file di *puro testo* (`txt`) o scritti con un linguaggio di *markup* molto semplice come MMD di cui si parlerà poco più avanti.

I file di ciascun coautore vanno archiviati in una cartella, eventualmente condivisi sul web. Il coordinatore provvede ad assemblare i vari contributi secondo l'indice prestabilito.

L'uso di markup semplici esalta la versatilità del sistema; attraverso opportuni convertitori è possibile ottenere dall'unica fonte così realizzata, e senza ulteriori rifacimenti, molteplici tipologie di documenti e differenti formattazioni, per esempio file in formato *pdf*, *html*, *opml*, *tex*, *rtf*, *docx*, fornendo nel contempo al lavoro una rifinitura accurata in ottica compositiva. Per approfondire questi aspetti si veda l'interessante articolo [17] sull'uso di Pandoc, di cui si è già parlato in questo capitolo; questo programma è definito dal suo autore come lo strumento più generale per la conversione di file da un formato ad un altro.

23.3.2.1 File `txt`

I file `txt`, se inseriti in strutture gerarchiche bene organizzate, rappresentano un modo sicuro e versatile per costruire e gestire un documento complesso. L'organizzazione è fondamentale sia in relazione alla struttura delle cartelle sia in relazione ai nomi da assegnare ai file.

Questi file sono immuni dalle differenze di piattaforma, resteranno sempre accessibili, purché si usino codifiche standard ben definite. La probabilità che restino accessibili per sempre è molto più alta rispetto a qualsiasi altro formato di

file di testo attualmente in circolazione. Inoltre essi sono leggeri e non necessitano di software particolari e costosi per essere letti e modificati.

Ovviamente immagini e tabelle (non gestite dal formato) andranno conservate in cartelle apposite, ordinatamente organizzate e riferite ai testi ed alle sezioni che le dovranno contenere.

Questi file andranno successivamente assemblati e composti (dal coordinatore dell'opera) con altri strumenti software da scegliere in funzione del sistema di lavoro adoperato per la produzione del documento finale.

23.3.2.2 File mmd e md

Per i file creati usando i markup MMD e MD valgono le stesse considerazioni per la gestione generale descritte per i file `txt` perché sono a tutti gli effetti file di puro testo

Ricordiamo che il markup *multimarkdown* è una evoluzione di quello denominato *markdown*; il primo possiede peculiarità più specificatamente rivolte alla produzione di testi in formato PDF, mentre il secondo è finalizzato al formato *html*. Nei rispettivi siti si legge quanto segue.

> **Markdown**, or MD, is a text-to-HTML conversion tool for web writers. Markdown allows you to write using an easy-to-read, easy-to-write plain text format, then convert it to structurally valid XHTML (or HTML). Thus, Markdown is two things: (1) a plain text formatting syntax; and (2) a software tool, written in Perl, that converts the plain text formatting to HTML.
>
> **MultiMarkdown**, or MMD, is a tool to help turn minimally marked-up plain text into well formatted documents, including HTML, PDF (by way of LaTeX), OPML, or OpenDocument (specifically, Flat OpenDocument (`.fodt`), which can in turn be converted into RTF, Microsoft Word, or virtually any other word-processing format). MMD is a superset of the Markdown syntax, originally created by John Gruber. It adds multiple syntax features (tables, footnotes, and citations, to name a few), in addition to the various output formats listed above (Markdown only creates HTML). Additionally, it builds in "smart" typography for various languages (proper left- and right-sided quotes, for example).

Con le loro sintassi specifiche *multimarkdown* o *markdown*, pur trattandosi di veri e propri linguaggi informatici, rende questi file facilmente intelligibili al cervello umano, molto più facilmente rispetto al caso che fossero scritti col markup *html*.

Per maggiori informazioni su questi linguaggi di markup si veda in:

`http://fletcherpenney.net/multimarkdown/`
e
`http://daringfireball.net/projects/markdown/`.

Questi file, se aperti con appositi software, possono essere visualizzati anche in modo sofisticato, cioè permettono di vedere il testo con formattazioni, immagini, tabelle, note a piè di pagina, eccetera.

Su Mac sono disponibili molte applicazioni che ne consentono la gestione. Tra queste ve ne sono due particolarmente interessanti nell'ottica di questa guida. Si tratta dei programmi Marked2 (`http://marked2app.com`) e MultimarkdownComposer (`http://multimarkdown.com`.

Il primo programma consente di visualizzare un file (che può anche essere salvato in in formato `txt`) usando *temi* pre-impostati o di propria creazione, o di esportarlo in diversi formati alternativi: *html, pdf, rtf, rtfd, docx, odt, md.* La sua utilità si esplica, ad esempio, nelle fasi iniziali di un'opera di gruppo, sulle bozze: consente di lavorare su file di puro testo senza perdere la possibilità di conferire loro un aspetto formattato (più o meno minimale) che ne facilita ulteriormente la leggibilità. Il file deve essere scritto con un altro editor, un qualsiasi editor di testi, dal più sofisticato al più spartano, e salvato in MMD, oppure MD, oppure `txt`. L'uso di MMD consente di attribuire a ciascun file un insieme di *metadati*, fattore che, in molti casi, riveste una significativa utilità pratica e gestionale.

Il secondo programma MultimarkdownComposer dispone anche di un proprio editor interno e offre anche la possibilità di esportare in formato `tex`; è un programma commerciale che si può usare solo su piattaforme Mac al costo di una decina di euro.

Attraverso l'accurata predisposizione di un albero di cartelle e file è possibile, già a questo punto ottenere ottimi risultati sia per lunghi documenti articolati sia per lavori meno impegnativi ma al tempo stesso strutturati, ad esempio un articolo.

23.3.3 Andare oltre: Scrivener

Il passo successivo, nel flusso metodologico della preparazione di un documento collettivo, è rappresentato dall'adozione di un *unico* software con cui:

- gestire ed organizzare tutti i file prodotti dai vari coautori (eventualmente con uno dei sistemi appena visti);
- organizzarli in parti, capitoli, sezioni, sottosezioni eccetera;
- modificare in corso d'opera l'indice generale dell'opera stessa.

Si tenga conto che è impossibile sintetizzare in poche pagine le funzioni disponibili in un software strutturato e versatile come Scrivener. In questa sede analizzeremo solo le caratteristiche legate ai temi di questa guida, rimandando chi desideri un approfondimento al sito web `https://www.literatureandlatte.com` e ad un *e-book* dedicato, reperibile nel sito `http://www.takecontrolbooks.com/scrivener-2`.

Scrivener è un applicativo in grado di amministrare all'interno di un unico file con estensione `.scriv` i molteplici contribuiti che danno luogo ad un'opera testuale; è uno strumento capace di gestire:

- le porzioni testo, le immagini, le tabelle, i weblink, le citazioni, il materiale di ricerca, eccetera, ivi compresi tutti i contributi inseriti adottando la sintassi MMD;

- le annotazioni correlate a ciascuno di essi;
- le annotazioni relative all'intera opera;
- le etichette descrittive associate a ciascuna porzione di testo che ne descrivono:
  - la tipologia gerarchica rispetto al sezionamento (parte, capitolo, paragrafo, sottoparagrafo e così via);
  - lo stato di avanzamento (To-do, first-draft, final-draft, o altro secondo le proprie necessità personali);
- gli elementi di studio e ricerca necessari ma che non entreranno a far parte della compilazione finale del documento.

Si tratta di un programma shareware, dal costo abbordabile di poche decine di euro, da pagare dopo il trentesimo giorno di uso effettivo; il denaro speso è ben ripagato dai vantaggi ricavabili con l'uso del programma.

Nasce come applicazione per *Mac OS X*, progettata da uno scrittore e specificatamente dedicata all'uso di chi deve scrivere libri, rapporti, manuali, articoli, copioni, eccetera. Grazie alla sua versatilità nel tempo ha acquisito una buona quota di mercato e si è aperta anche alle piattaforme Windows e Linux.

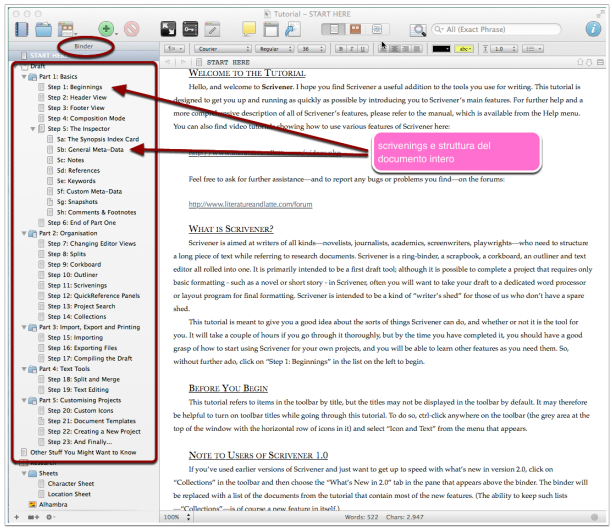
Per questa guida gli aspetti di interesse sono rappresentati dalle capacità di:

- gestire file di testo provenienti da altre applicazioni (ad esempio Word, Writer, Pages,...),
- gestire il *multimarkdown*,
- stampare in formato PDF il testo marcato in *multimarkdown* attraverso una compilazione che si appoggia a LaTeX,
- generare file `tex` originati da testi marcati in *multimarkdown*.

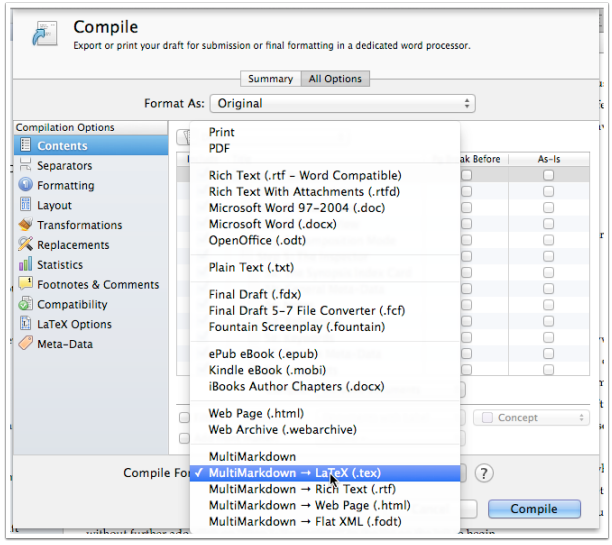
In Scrivener un documento è suddiviso in tanti frammenti di testo. Ogni frammento di testo è un file; ciò è utile per il lavoro in senso generale e fondamentale nel caso in cui le bozze provengano da file di testo redatti ed archiviati nei modi descritti in precedenza. Questi frammenti di testo prendono il nome di *scrivening*. Essi vengono ordinati nel raccoglitore (*binder*) dall'autore o, nel caso di più autori, dal coordinatore. Il *binder* è il gestore del documento, contiene e consente l'ordinamento dei vari scrivening, compresi quei file di lavoro che non entreranno a far parte del documento finale compilato (vedi figura 23.1).

Rispetto all'uso di comuni text editor e word processor, come per esempio Word, Pages, TextEdit, Writer delle suite Libre Office o Open Office, ciò agevola la gestione di un testo lungo ed articolato da sezionare in parti, capitoli e paragrafi, numerati e riferiti ad un indice generale unico, e facilita l'elaborazione dei contenuti in modo selettivo sempre lavorando all'interno di un unico file file.scriv. In questo modo si può lavorare con grande comodità pratica di gestione del file a capitoli e paragrafi non necessariamente consecutivi e/o in sequenza.

Una funzione molto efficace, specie in caso di lavoro collaborativo e multipiattaforma, consente di importare un file di Word dividendolo, se serve, in tanti scrivening. Ciò, però, avviene solo se chi lo ha redatto ha avuto l'accortezza di



(a) la finestra principale di scrivener



(b) I formati di esportazione

Figura 23.1: La finestra di compilazione di Scrivener

anteporre uno o più segni `#` al titolo di ciascun capitolo o paragrafo. Questo aspetto evidenzia, fra l'altro, la necessità di redigere preventivamente le linee guida operative di lavoro comune.

Con eguale versatilità si possono trascinare direttamente nel *binder* i file testuali dei vari coautori. In questo caso ogni documento trascinato rappresenterà uno scrivening.

Gli scrivening risiedono nelle cartelle del binder; queste possono essere inserite una dentro l'altra a formare una specie di albero strutturato. In tal caso si genereranno automaticamente, in fase di compilazione, i capitoli, i paragrafi e i sottoparagrafi. Nella stampa da MMD in PDF e nell'esportazione in LaTeX si originano così i necessari comandi `\chapter`, `\section`, `\subsection`, `\subsubsection`, eccetera.

Nella versione per Mac sono disponibili altre funzionalità che verrano descritte nelle sezioni seguenti.

23.3.3.1 Compilazione del documento finale in PDF

Dal punto di vista operativo la situazione ottimale si presenta quando tutti i contributi testuali sono redatti fin dall'origine con sintassi MMD. Altrimenti, per poter usufruire della caratteristiche descritte nei successivi paragrafi, toccherà al coordinatore provvedere all'applicazione del *multimarkdown* agli scrivening che ne sono sprovvisti.

Il documento finale, costituito dagli *scrivenings* scritti in MMD e raccolti nel binder, va compilato in Scrivener per poter essere stampato o esportato in altri formati; qui ci limitiamo alle formattazioni che afferiscono al mondo LaTeX (vedi figura 23.1 b).

Possiamo percorrere due vie.

1. Compiliamo direttamente da MMD per ottenere un documento in PDF;
2. Compiliamo da MMD generando un file con estensione `tex` da gestire con applicativi specifici per LaTeX.

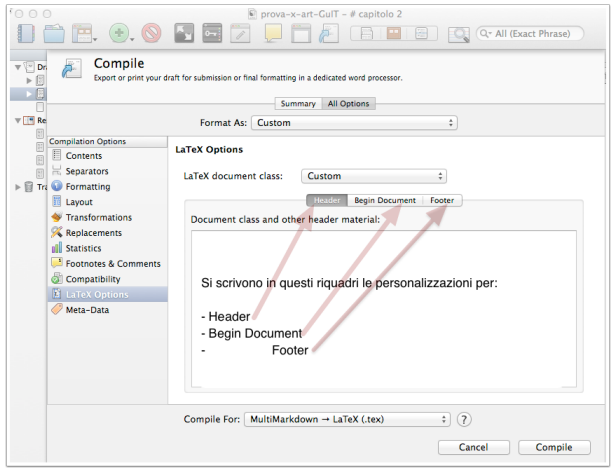
Avendo deciso di *stampare* direttamente da MMD in PDF selezioniamo la voce:

```
multimarkdown --> pdf
```

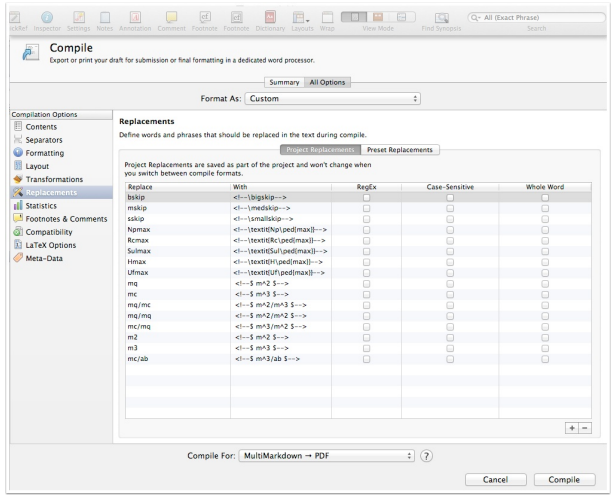
come mostrato nella figura 23.3a. Scrivener eseguirà tre compilazioni tramite LaTeX per costruire indice, glossario, bibliografie, eccetera, restituendo un documento PDF formattato secondo le scelte impostate nella finestra 'LaTeX Options': *memoir* (book), *article*, oppure *Custom*. Quest'ultima opzione presuppone che i tre campi: 'Header', 'Begin Document' e 'Footer' vengano riempiti con le informazioni consuete necessarie per lavorare in LaTeX (vedi figure 23.1 e 23.2a).

Per l'uso pratico del sistema di lavoro, il documento composto in PDF da Scrivener, via LaTeX, è comodo:

- nelle fasi intermedie di lavoro, per la revisione delle bozze (che così avranno già una prima impaginazione in stile LaTeX);



(a) personalizzare il documento finale con LaTeX



(b) uso di 'Replacement' insieme ai commenti *html*

Figura 23.2: La compilazione da Scrivener

- per chi non conosce (o conosce ancora poco) LaTeX ma desidera sfruttarne alcune potenzialità.

23.3.3.2 Compilazione del documento finale in file `tex`

L'alternativa all'output diretto in PDF è fornita dalla compilazione in uno o più file `tex` da gestire successivamente con un applicativo specifico per LaTeX (se ne veda la pagina di impostazione nella figura 23.3 b). Compilare i vari scrivening in tanti file `tex` diventa efficace se si prevede di organizzare il lavoro in LaTeX adoperando `\include` e/o `\input`. Il file ottenuto da Scrivener è già compilabile (da LaTeX), così com'è, oppure può essere *manipolato* impostando da capo la classe del documento, il preambolo e i pacchetti in relazione alle capacità ed alle necessità. La derivazione diretta dalla compilazione via Scrivener dà origine ad una *grammatica* LaTeX non sempre ottimale, sicché non guasta (sebbene non sia sempre necessario) procedere a un'attenta revisione preventiva per rifinire il codice.

23.3.4 Commenti in stile *html*

Lavorando all'interno di Scrivener con markup MMD è possibile servirsi di sintassi e comandi abituali di LaTeX. Per farlo si ricorre ai *tag* del tipo `<!-- testo -->` secondo la sintassi tipica adoperata per annotare il codice o la pagina scritti in *html*.

Supponiamo, ad esempio, di voler apporre una nota a margine del testo. All'interno dello scrivening, in corrispondenza del punto in cui va inserita la nota, scriviamo il codice LaTeX come se fosse un commento *html*, così:

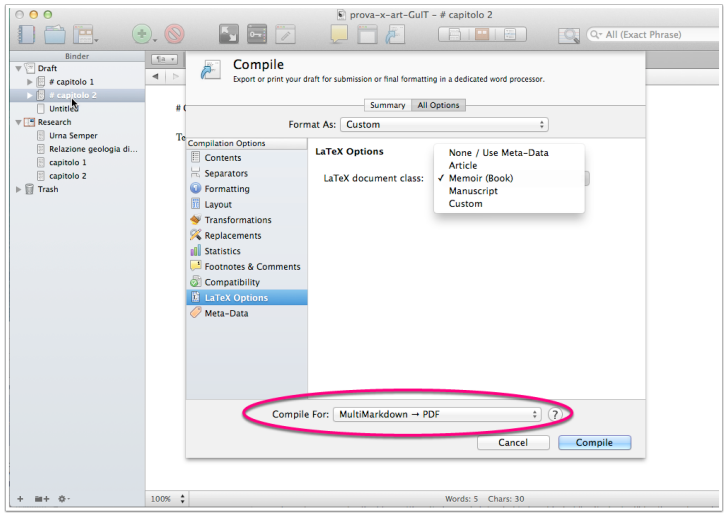
```
<!--\marginpar{testo della nota a margine}-->
```

Nota a margine del testo predisposta in Scrivener.

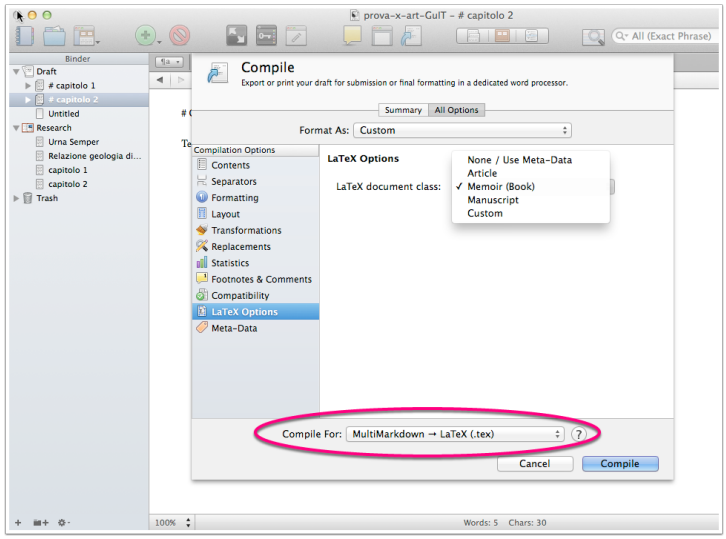
Al termine della compilazione lanciata da Scrivener comparirà la nota nel punto voluto[10]. Con questo semplice metodo si possono includere molti tratti di codice LaTeX: scatole di testo, anche lunghe, o altre specifiche azioni quali, ad esempio, formule matematiche o testi particolari.

Un altro aspetto che merita di essere segnalato e molto utile sul piano pratico è dato dall'utilizzo congiunto dei commenti *html* e della funzione 'Replacements' che appare tra le opzioni impostabili nei dialoghi della compilazione (vedi figura 23.2b). Si possono impostare sostituzioni per semplificare la digitazione dei testi in *mmd*. Ipotizzando, ad esempio, di dover scrivere molte unità di misura secondo le prescrizioni del *Sistema Internazionale*, può diventare fastidioso dover ricorrere spesso al tag *html*. È più semplice impostare la sostituzione di un testo *abbreviato* e comodo da digitare (supponiamo *mq*) via 'Replacements'. Per farlo, in corrispondenza della colonna 'Replace' scriviamo il testo abbreviato ('mq' oppure 'm2'), e nella colonna 'with' immettiamo il testo corretto racchiuso nel commento *html*, che in questo caso potrà essere:

[10] La nota a margine sarà stampata nel PDF in caso di compilazione diretta. Se invece si esporta in un file *tex*, il codice LaTeX sarà posto correttamente nel punto desiderato.



(a) compila da MMD a PDF



(b) compila da MMD a LaTeX

Figura 23.3: La compilazione da Scrivener in MMD o LaTeX

```
<!-- \ensuremath{\mathrm{m^2} -->
```

In tutto il documento potremo scrivere l'abbreviazione *mq* ogni volta che ci servirà sapendo che nel documento stampato (PDF) o nel file `tex` l'unità di misura apparirà scritta nel modo corretto (m^2), con l'esponente 2. Con un po' di esercizio si possono comporre sintassi molto più complicate.

Questo metodo è utilizzabile con profitto, per esempio, per mettere in posizione i comandi di spaziatura verticale tra i capoversi. In questo modo digitando, in Scrivener, le lettere *bskip* ed avendo impostato nel 'Replacement' che ad esse corrisponda `<!-- \bigskip -->`, si ottiene:

- nel testo stampato lo spazio maggiore desiderato;
- nel nel file `tex` l'inserimento del comando `\bigskip`.

Come molte applicazioni complesse e versatili l'uso genera la crescente conoscenza delle potenzialità e congiuntamente la possibilità di sviluppare usi particolarmente adatti alle proprie necessità.

In ogni caso, Scrivener non può essere visto o considerato come uno *shell editor* per LaTeX. La distanza da TeXShop, TeXWorks, Texpad (per citare solo alcuni applicativi specialistici) è molta ed incolmabile. Tuttavia rappresenta uno strumento valido per avvicinarsi *in punta di piedi* a questo mondo. Per ottenere in breve tempo risultati soddisfacenti è sufficiente imparare *multimarkdown*, impegno decisamente più semplice e meno dispendioso in termini di tempo e fatica. Certo, iniziando con meno fatica, rispetto all'approccio diretto a LaTeX, la voglia di migliorare, visti da vicino i risultati, non potrà fare altro che spingere molti utenti verso la ricerca di approfondimento e quindi verso l'apprendimento di LaTeX e dei suoi strumenti specifici.

Sul forum di Scrivener, nella categoria 'multimarkdown', si trovano spunti ed esperienze interessanti. Tra questi una guida per principianti (in inglese), dal titolo *A toad's guide to using Scrivener, MultiMarkDown and LaTeX*, scaricabile all'indirizzo web `http://www.literatureandlatte.com/forum/viewtopic.php?f=21&t=17239&start=0`.

23.4 Conclusioni

Come si vede i processi per importare in documenti LaTeX il contenuto di documenti in formati diversi sono piuttosto complessi, non tanto perché ci siano delle operazioni difficili da eseguire, quanto perché a tutt'oggi non esiste un modo solo, unificato e corretto per mettere in un documento LaTeX delle informazioni tratte da altri documenti che sono privi di tutte le informazioni necessarie. Nessun programma, al di là di alcune impostazioni di default, può inventarsi che cosa aveva in mente chi ha scritto originariamente il documento iniziale.

Va aggiunto che gli utenti dei word processor di solito non usano quel minimo di mark-up che programmi come Microsoft Word o OpenOffice.org Writer consentono, per esempio di scegliere gli header dall'apposito menù. Quegli

header sono necessari per eseguire automaticamente l'indice del documento, ma la maggior parte dei documenti sono privi di indice e gli utenti preferiscono scrivere l'intestazione e poi cambiare il font e la serie con semplici click del mouse, piuttosto che cliccare su un menù a discesa che offre una vasta scelta di header ma con poche spiegazioni di che cosa significhino quei termini. Risultato: anche il miglior convertitore fa fatica a convertire in formato adatto a LaTeX, per cui il compositore deve consumare un tempo notevole per correggere quello che il convertitore non ha saputo fare adeguatamente per trasformare un documento di scarsa qualità in un documento che vorrebbe essere di qualità assai superiore, ma non può proprio per mancanza di informazione adeguata.

Quando, quindi, bisogna produrre un documento di qualità con LaTeX partendo da numerosi contributi scritti con word processor diversi, il compositore deve armarsi di santa pazienza anche se ricorre a programmi di conversione che, però, ricordiamolo bene, non possono fare l'impossibile.

Capitolo 24

Simbologia matematica e fisica

24.1 Unità di misura del Sistema Internazionale

Le unità fondamentali del Sistema Internazionale sono raccolte nella tabella 24.1; secondo le norme internazionali gli angoli piani e solidi sono considerati "quantità derivate adimensionate"; infatti le norme affermano che "le unità *radiante* e *steradiante* devono essere considerate come unità derivate adimensionate che possono essere usate od omesse nelle espressioni delle unità derivate". È per questo che più avanti per alcune grandezze fisiche saranno indicate *fra parentesi* le unità di misura contenenti anche i radianti o gli steradianti in quei casi in cui il loro uso consente di distinguere specie fisiche diverse ma apparentemente equidimensionate.

Grandezza fisica	Unità	Simbolo
lunghezza	metro	m
massa	kilogrammo	kg
tempo	secondo	s
corrente elettrica	ampere	A
temperatura termodinamica	kelvin	K
quantità di sostanza	mole	mol
intensità luminosa	candela	cd

Tabella 24.1: Unità fondamentali

Tutte queste unità, nonché quelle delle tabelle successive, possono essere precedute dai prefissi decimali raccolti nella tabella 24.2. Si ricorda che i prefissi vanno usati isolatamente (in passato non era infrequente osservare il prefisso *millimicro* al posto del prefisso corretto *nano*). Quando l'unità di misura con

prefisso è elevata ad un esponente, questo si intende applicato all'unità completa di prefisso: $3\,\mathrm{cm}^3$ indica un volume di $3\,(10^{-2}\mathrm{m})^3 = 3\times10^{-6}\,\mathrm{m}^3$ e non un volume di $3\times10^{-2}\,\mathrm{m}^3$.

Quando si parla di byte (simbolo B) si usano i prefissi binari; questi sono riportati nella tabella 24.3; essi sono legali dal 1998 e dovrebbero venire sempre usati quando si parla di quantità "informatiche", come le capienze dei dischi, o le dimensioni di certe memorie, e simili. Si noti che per l'unità kibi (kilo binario) il simbolo comincia con una K maiuscola; non è un errore: tutti i prefissi binari cominciano con lettere maiuscole incluso quello che ricorda il prefisso kilo; lo si ricorda solamente, ma Ki sta per 1024, mentre k sta per 1000; sono evidentemente due cose diverse.

Prefisso	Valore	Simbolo	Prefisso	Valore	Simbolo
yotta	10^{24}	Y	deci	10^{-1}	d
zetta	10^{21}	Z	centi	10^{-2}	c
exa	10^{18}	E	milli	10^{-3}	m
peta	10^{15}	P	micro	10^{-6}	µ
tera	10^{12}	T	nano	10^{-9}	n
giga	10^{9}	G	pico	10^{-12}	p
mega	10^{6}	M	femto	10^{-15}	f
kilo	10^{3}	k	atto	10^{-18}	a
etto	10^{2}	h	zepto	10^{-21}	z
deca	10^{1}	da	yocto	10^{-24}	y

Tabella 24.2: Prefissi decimali

Prefisso	Valore	Simbolo
kibi	2^{10}	Ki
mibi	2^{20}	Mi
gibi	2^{30}	Gi
tebi	2^{40}	Ti
pebi	2^{50}	Pi
exbi	2^{60}	Ei

Tabella 24.3: Prefissi binari

Grandezza	Unità	Simbolo
attenuazione, guadagno	decibel	dB
attenuazione, guadagno	neper	Np
intervallo di frequenza	ottava	ott
intervallo di frequenza	decade	dec

Tabella 24.4: Unità logaritmiche

Per quanto riguarda le cosiddette unità logaritmiche, sono codificate quelle della tabella 24.4; si ricorda che i nomi che vengono dati a queste unità servono solo a ricordare quale base è stata usata per il calcolo del logaritmo. Si richiama l'attenzione anche sulla corretta scrittura dei simboli dB e Np, che invece si vedono così spesso scritti in modo errato. Per quanto riguarda le unità di attenuazione e di guadagno si usano i logaritmi decimali per i decibel, o naturali per i neper, ed in più si hanno definizioni diverse a seconda che il rapporto di cui si calcola il logaritmo sia eseguito fra grandezze di potenza o energia, oppure fra grandezze di campo:

$$\alpha = 10 \log_{10} \frac{P_1}{P_2} \qquad \text{oppure} \qquad \alpha = \frac{1}{2} \log_{\mathrm{e}} \frac{P_1}{P_2}$$

dove P_1 e P_2 sono potenze, oppure

$$\alpha = 20 \log_{10} \frac{V_1}{V_2} \qquad \text{oppure} \qquad \alpha = \log_{\mathrm{e}} \frac{V_1}{V_2}$$

dove V_1 e V_2 sono tensioni.

Per gli intervalli di frequenza si usano i logaritmi in base 2 per le ottave, o in base 10 per le decadi

$$I = \log_2 \frac{f_2}{f_1} \qquad \text{oppure} \qquad I = \log_{10} \frac{f_2}{f_1}$$

È stato necessario introdurre molte altre unità per le grandezze fisiche derivate, al fine di evitare di dover usare lunghi elenchi di unità fondamentali elevate a potenze insolite, che sarebbe fra l'altro troppo complicato ricordare; queste unità derivate sono elencate nella tabella 24.5.

Un cenno particolare merita il *litro* perché sono leciti ben due simboli per questa unità: l, L; viene usato (impropriamente) anche ℓ, ma non è 'legale'; invece questo simbolo, se fosse legale, costituirebbe una scelta quanto mai opportuna, perché evita ogni possibile confusione della l minuscola con la cifra 1, e quella della L maiuscola con la cifra 4.

Le unità ammesse sono riportate nella tabella 24.6.

Infine sono ancora *tollerate* o *temporaneamente accettate* alcune altre unità *in via di estinzione*; quelle che si sono già estinte (come l'atmosfera, il quintale, il millimetro di mercurio — ammesso solo in campo medico —, il poise, eccetera) non sono nemmeno elencate proprio per evitare che possa venire la tentazione di usarle ancora. Le convenzioni internazionali, a cui l'Italia aderisce, faranno sparire in un prossimo futuro anche queste unità tollerate, che sono elencate nella tabella 24.7.

Nelle tabelle 24.1–24.7 si notano delle assenze vistose, oltre a quelle già segnalate; in particolare mancano tutte le unità CGS, dagli erg alle dine, dai gauss agli oersted, tanto per citare quelle più comuni; non sono accettati il kilogrammo-forza, il micron, il carato metrico, la caloria, l'atmosfera. Si notano inoltre le assenze dei simboli cc, mc, mmc, mq, che sono scorrettamente tanto

Grandezza fisica	Unità	Simbolo
angolo piano	radiante	rad
angolo solido	steradiante	sr
frequenza	hertz	Hz
forza	newton	N
pressione	pascal	Pa
lavoro, energia	joule	J
potenza	watt	W
carica elettrica	coulomb	C
tensione elettrica	volt	V
capacità elettrica	farad	F
resistenza elettrica	ohm	Ω
conduttanza elettrica	siemens	S
flusso di induzione magnetica	weber	Wb
induzione magnetica	tesla	T
induttanza	henry	H
flusso luminoso	lumen	lm
illuminamento	lux	lx
attività di un radionuclide	becquerel	Bq
dose assorbita	gray	Gy
equivalente di dose	sievert	Sv
attività catalitica	katal	kat

Tabella 24.5: Unità derivate

comuni in alcune scienze; al loro posto vanno usati i simboli corretti cm^3, m^3, mm^3, m^2. Una volta, ai tempi della dattilografia, era piuttosto laborioso inserire gli esponenti; oggi con i sistemi di videoscrittura, in particolare di composizione tipografica, come LaTeX, gli esponenti non sono più un problema, quindi quelle abbreviazioni errate, tollerate per necessità di cose fino a trenta anni fa, oggi non sono più accettabili.

24.2 Simboli matematici nelle scienze

In questo paragrafo sono raccolti i simboli matematici più comuni che si impiegano nelle scienze e nella fisica; essi sono ispirati tra l'altro alle norme CNR-UNI 10002 e alle norme CEI 24-1, ma, quando queste norme erano in conflitto, è stata operata una scelta arbitraria se usare il simbolo proposto da una norma piuttosto che quello indicato dall'altra, oppure se indicarli entrambi. È stato seguito questo criterio anche per la scelta dei simboli che si ritengono meno frequenti e che non sono stati inseriti nella tabella 24.8.

Come al solito l'elenco non è e non può essere completo, ma può servire da guida o modello per preparare un analogo elenco qualora si facesse uso di una

Grandezza fisica	Unità	Simbolo	Equivalenza
angolo piano	grado sessagesimale	°	$1° = \pi\,\mathrm{rad}/180$
angolo piano	minuto sessagesimale	′	$1′ = \pi\,\mathrm{rad}/10\,800$
angolo piano	secondo sessagesimale	″	$1″ = \pi\,\mathrm{rad}/648\,000$
angolo piano	gon o grado centesimale	gon	$1\,\mathrm{gon} = \pi\,\mathrm{rad}/200$
angolo piano	giro	giro	$1\,\mathrm{giro} = 2\pi\,\mathrm{rad}$
area	ara	a	$1\,\mathrm{a} = 100\,\mathrm{m}^2$
area	ettaro	ha	$1\,\mathrm{ha} = 10\,000\,\mathrm{m}^2$
volume	litro	l, L	$1\,\mathrm{L} = 1\,\mathrm{dm}^3$
tempo	minuto	min	$1\,\mathrm{min} = 60\,\mathrm{s}$
tempo	ora	h	$1\,\mathrm{h} = 3600\,\mathrm{s}$
tempo	giorno	d	$1\,\mathrm{d} = 86\,400\,\mathrm{s}$
massa	tonnellata	t	$1\,\mathrm{t} = 1000\,\mathrm{kg}$
massa	unità di massa atomica	u	$1\,\mathrm{u} = 1{,}660\,57 \times 10^{-27}\,\mathrm{kg}$
lavoro, energia	elettronvolt	eV	$1\,\mathrm{eV} = 1{,}602\,19 \times 10^{-19}\,\mathrm{J}$
lavoro, energia	kilowattora	kW h	$1\,\mathrm{kW\,h} = 3{,}6\,\mathrm{MJ}$
carica elettrica	amperora	A h	$1\,\mathrm{A\,h} = 3600\,\mathrm{C}$
temperatura Celsius	grado Celsius	°C	$1\,°\mathrm{C} = 1\,\mathrm{K}$ ma differisce lo zero della scala: $t = T - 273{,}15\,\mathrm{K}$

Tabella 24.6: Unità di misura legalmente ammesse

Grandezza fisica	Unità	Simbolo	Equivalenza
lunghezza	miglio marino	miglio marino	$1\,\text{miglio marino} = 1852\,\text{m}$
lunghezza	ångström	Å	$1\text{Å} = 10^{-10}\,\text{m}$
area	ara	a	$1\,\text{a} = 100\,\text{m}^2$
area	ettaro	ha	$1\,\text{ha} = 10\,000\,\text{m}^2$
area	barn	b	$1\,\text{barn} = 10^{-28}\,\text{m}^2$
pressione	bar	bar	$1\,\text{bar} = 100\,\text{kPa}$
velocità	nodo	nodo	$1\,\text{nodo} = (4{,}63/9)\,\text{m/s}$ $1\,\text{nodo} = 1\,\text{miglio marino/h}$
accelerazione	gal	Gal	$1\,\text{Gal} = 1\,\text{cm/s}^2$
attività di radionuclide	curie	Ci	$1\,\text{Ci} = 3{,}7 \times 10^{10}\,\text{Bq}$
esposizione a raggi X o γ	roentgen	R	$1\,\text{R} = 2{,}58 \times 10^{-4}\,\text{C/kg}$
dose assorbita	rad	rad, rd	$1\,\text{rd} = 1\,\text{cGy}$
equivalente di dose	rem	rem	$1\,\text{rem} = 1\,\text{cSv}$

Tabella 24.7: Unità di misura temporaneamente accettate

simbologia fisico-matematica piuttosto elaborata.

Nella tabella 24.8 le lettere a e b sono due numeri reali qualsiasi, i, j, k, n sono numeri interi, z, s sono variabili o numeri complessi, x, y, (talvolta anche z), e t sono variabili reali, D è un dominio, A, B, C, P sono punti del piano o dello spazio. La colonna intestata **Simbolo** contiene il segno grafico del simbolo, oppure un'espressione che ne fa uso.

24.3 Nomenclatura

È praticamente impossibile fare un elenco di tutti nomi delle grandezze che vengono usate in ogni scienza, dalla fisica alla medicina, dall'elettronica alla geologia. Si ritiene però cosa utile riprendere l'elenco del prospetto IV della norma CNR-UNI 10003, ampliandolo un poco e aggiungendovi il simbolo (o una scelta di simboli) che sono comunemente accettati in ogni scritto scientifico, senza che sorga la necessità di compilare un elenco delle grandezze e dei simboli usati.

Nel compilare la tabella 24.9 si è tratta ispirazione dalle norme CNR-UNI, dalle norme CEI, dal fascicolo CEI di nomenclatura nucleare, dal documento sulla nomenclatura pubblicato dalla Società Internazionale di Fisica, senza inventare nulla, ma operando solo delle scelte fra le grandezze o i simboli che sono stati inclusi o esclusi nella tabella.

Fra parentesi, nella colonna delle unità di misura, vi sono delle indicazioni ulteriori che comprendono anche i radianti o altre unità come i neper o i cicli,

quando è parso che l'introduzione di queste unità accessorie rendesse più chiara la differenza fra grandezze di specie diversa ma apparentemente equidimensionate.

La tabella 24.9, come detto sopra, è certamente incompleta, ma rappresenta comunque un modello da imitare qualora fosse necessario fare un elenco delle grandezze e dei simboli usati nel documento.

Tabella 24.8: Simboli matematici

Simbolo	**Significato**	**Note**
,	virgola decimale	Non usare il punto per separare la parte intera dalla parte decimale. Non usare nemmeno altri separatori tra i gruppi di tre cifre prima e dopo la virgola. Il punto decimale si può usare solo scrivendo in inglese, ma i separatori fra gruppi di tre cifre devono essere assenti lo stesso. In entrambi i casi il separatore può essere costituito da uno spazio fine non separabile
∞	infinito	
π	$\pi = 3{,}141\,592\ldots$	Se il font lo consente, sarebbe desiderabile stampare la costante matematica π con un carattere non inclinato
e	$\mathrm{e} = 2{,}718\,281\ldots$	Va scritta in tondo, se non altro per distinguerla dalla carica elementare e; ma, come base degli esponenziali neperiani, sarebbe preferibile trattarla sempre come un operatore.
γ	$\gamma = 0{,}577\,215\ldots$	Vale la stessa osservazione fatta per π
i , j	unità immaginaria, operatore di rotazione	Va scritta in tondo come gli altri operatori

continua

Continua tabella 24.8

Simbolo	Significato	Note
...	omissione	Si usa sia nel significato di *elementi omessi* sia in quello di *eccetera*
x, y, z	coordinate cartesiane	x: larghezza, y: profondità, z: altezza
ϱ, φ, z	coordinate cilindriche	ϱ: distanza dall'asse, φ: longitudine, z: altezza
ϱ, φ, ϑ	coordinate sferiche	ϱ: distanza radiale, φ: longitudine, ϑ: colatitudine
$a = b$	uguale	
$a \neq b$	diverso	
$a \equiv b$	identico	
$\mathrm{e} \approx 2{,}718$	uguale a circa	
$a \sim b$	proporzionale	Si può usare anche $a \propto b$
$a \leftrightarrow b$	equivalente	
$a > b$	maggiore	
$a < b$	minore	
$a \geq b$	maggiore o uguale	
$a \leq b$	minore o uguale	
$a \gg b$	molto maggiore	
$a \ll b$	molto minore	
$a \to b$	tendente	
$a \simeq b$	asintoticamente uguale	
$a \triangleq b$	corrispondente	Si usa nelle indicazioni di scala dei diagrammi: per esempio $1\,\mathrm{cm} \triangleq 10\,\mathrm{V}$
$a \div b$	intervallo	Si usa nel senso di "da a a b"
$a + b$	somma	

continua

Continua tabella 24.8

Simbolo	Significato	Note
$a - b$	sottrazione	
ab , $a \cdot b$	moltiplicazione	Non usare altri simboli quando gli operandi sono indicati mediante lettere
$1{,}5 \times 2{,}3$	moltiplicazione	Non usare altri simboli quando gli operandi sono entrambi numerici
$\begin{cases} 1{,}5\,a \\ 1{,}5 \cdot a \end{cases}$	moltiplicazione	Gli operandi numerici precedono sempre quelli letterali
a/b , $\dfrac{a}{b}$	divisione	Le due simbologie possono essere mescolate; usare le parentesi per isolare le singole operazioni ed evitare ambiguità; per esempio $\dfrac{(a/b)+1}{(a/b)+(b/a)}$
$a \bmod b$	modulo	resto della divisione a/b con quoziente intero; è sempre $a \bmod b = \dfrac{a}{b} - \left\lfloor \dfrac{a}{b} \right\rfloor$ quindi $0 \leq a \bmod b < 1$
a^b	elevazione a potenza	
$\sqrt[b]{a}$	estrazione di radice	Non usare né $\sqrt[b]{}a$ né $\sqrt[b]{}(a)$; se $b = 2$, b viene omesso
$\lvert a \rvert$	valore assoluto	
$\sum_{i=1}^{n} a_i$	somma	

continua

Continua tabella 24.8

Simbolo	Significato	Note
$\prod_{i=1}^{n} a_i$	prodotto	
$n!$	fattoriale	
$\binom{n}{m}$	coefficiente binomiale	$\frac{n(n-1)\cdots(n-m+1)}{1\times 2\times\cdots m}$
$f(x)$	funzione	
$\log_b x$	logaritmo in base b	
$\log x$	logaritmo in base 10	
$\ln x$, $\log_e x$	logaritmo naturale o neperiano	
$\mathrm{lb}\, x$, $\log_2 x$	logaritmo in base 2	
e^x , $\exp x$	esponenziale	In questa e nelle funzioni successive scritte in caratteri tondi l'argomento non necessita di parentesi quando è composto da un solo elemento letterale o numerico
$\sin x$	seno	
$\cos x$	coseno	
$\tan x$	tangente	
$\cot x$	cotangente	
$\sinh x$	seno iperbolico	
$\cosh x$	coseno iperbolico	
$\tanh x$	tangente iperbolica	
$\coth x$	cotangente iperbolica	
$\arcsin x$	arcoseno	
$\arccos x$	arcocoseno	

continua

Continua tabella 24.8

Simbolo	Significato	Note
$\arctan x$	arcotangente	
$\operatorname{arccot} x$	arcocotangente	
$\operatorname{arsinh} x$	arcoseno iperbolico	
$\operatorname{arcosh} x$	arcocoseno iperbolico	
$\operatorname{artanh} x$	arcotangente iperbolica	
$\operatorname{arcoth} x$	arcocotangente iperbolica	
$K(k)$	integrale ellittico completo di prima specie	$K(k) = \int_0^{\pi/2} \frac{\mathrm{d}\vartheta}{\sqrt{1 - k^2 \sin^2\vartheta}}$
$F(\varphi, k)$	integrale ellittico incompleto di prima specie	$F(\varphi,\, k) = \int_0^{\varphi} \frac{\mathrm{d}\vartheta}{\sqrt{1 - k^2 \sin^2\vartheta}}$
$E(\varphi, k)$	integrale ellittico incompleto di seconda specie	$E(\varphi, k) = \int_0^{\varphi} \sqrt{1 - k^2 \sin^2\vartheta}\, \mathrm{d}\vartheta$
$\Pi(n; \varphi, k)$	integrale ellittico incompleto di terza specie	$\Pi(n; \varphi, k) = \int_0^{\varphi} \frac{\mathrm{d}\vartheta}{(1 - n \sin^2\vartheta)\sqrt{1 - k^2 \sin^2\vartheta}}$
$\varphi(x, k)$	amplitudine	L'amplitudine è legata all'integrale ellittico di prima specie dalla relazione $x = F(\varphi,\, k)$
$\operatorname{sn}(x, k)$	seno ellittico	$\operatorname{sn}(x, k) = \sin\varphi$
$\operatorname{cn}(x, k)$	coseno ellittico	$\operatorname{cn}(x, k) = \cos\varphi$
$\begin{cases}\operatorname{dn}(x,\, k)\\ \Delta(\varphi)\end{cases}$	delta amplitudine	$\Delta(\varphi) = \sqrt{1 - k^2 \sin^2\varphi}$
$o(x)$	ordine di infinito o infinitesimo	Se $y = o(x)$ allora $\lim y/x = 0$
$O(x)$	ordine di infinito o infinitesimo	Se $y = O(x)$ allora $\lvert \lim y/x \rvert < \infty$
$\Gamma(z)$	funzione gamma	$\Gamma(z) = \int_0^{\infty} t^{z-1}\, \mathrm{e}^{-t}\, \mathrm{d}t$

continua

Continua tabella 24.8

Simbolo	Significato	Note
$(a)_n$	simbolo di Pochhammer	$(a)_n = \dfrac{\Gamma(a+n)}{\Gamma(a)}$
$\operatorname{erf}(z)$	funzione d'errore	$\operatorname{erf}(z) = \dfrac{2}{\sqrt{\pi}} \displaystyle\int_0^z \mathrm{e}^{-t^2}\,\mathrm{d}t$
$\operatorname{erfc}(z)$	funzione complementare d'errore	$\operatorname{erfc}(z) = 1 - \operatorname{erf}(z)$
$\mathrm{C}(z)$	integrale di Fresnel	$\mathrm{C}(z) = \displaystyle\int_0^z \cos(\pi t^2/2)\,\mathrm{d}t$
$\mathrm{S}(z)$	integrale di Fresnel	$\mathrm{S}(z) = \displaystyle\int_0^z \sin(\pi t^2/2)\,\mathrm{d}t$
$\mathrm{Si}(z)$	seno integrale	$\mathrm{Si}(z) = \displaystyle\int_0^z \frac{\sin t}{t}\,\mathrm{d}t$
$\mathrm{Ci}(z)$	coseno integrale	$\mathrm{Ci}(z) = \gamma + \ln z + \displaystyle\int_0^z \frac{\cos t - 1}{t}\,\mathrm{d}t$
$E_1(z)$	esponenziale integrale	$E_1(z) = \displaystyle\int_z^\infty \frac{\mathrm{e}^{-t}}{t}\,\mathrm{d}t$
$\mathrm{Ei}(x)$	esponenziale integrale	$\mathrm{Ei}(x) = \displaystyle\int_{-\infty}^x \frac{\mathrm{e}^{-t}}{t}\,\mathrm{d}t$
$\mathrm{li}(x)$	logaritmo integrale	$\mathrm{li}(x) = \displaystyle\int_0^x \frac{\mathrm{d}t}{\ln t} = \mathrm{Ei}(\ln x)$
$\zeta(s)$	funzione Zeta di Riemann	$\zeta(s) = \displaystyle\sum_{k=1}^{\infty} k^{-s}$
$\delta(t)$	distribuzione di Dirac	
$\mathrm{u}(t)$	gradino unitario	$\mathrm{u}(t) = \begin{cases} 0 & \text{per } t < 0 \\ 1/2 & \text{per } t = 0 \\ 1 & \text{per } t > 0 \end{cases}$
δ_{ij}	simbolo di Kronecker	$\delta_{ij} = \begin{cases} 0 & \text{per } i \neq j \\ 1 & \text{per } i = j \end{cases}$
$J_\nu(z)$	funzione di Bessel di prima specie	
$Y_\nu(z)$	funzione di Bessel di seconda specie	

continua

Continua tabella 24.8

Simbolo	Significato	Note
$H_\nu^{(1)}(z)$	funzione di Hankel di prima specie	$H_\nu^{(1)}(z) = J_\nu(z) + \mathrm{i}\, Y_\nu(z)$
$H_\nu^{(2)}(z)$	funzione di Hankel di seconda specie	$H_\nu^{(2)}(z) = J_\nu(z) - \mathrm{i}\, Y_\nu(z)$
$I_\nu(z)$	funzione di Bessel modificata di prima specie	
$K_\nu(z)$	funzione di Bessel modificata di seconda specie	
$\mathrm{ber}_\nu(x)$	prima funzione di Kelvin di prima specie	$\mathrm{ber}_\nu(x) = \mathbf{Re}\left[J_\nu\left(x\, \mathrm{e}^{3\pi\,\mathrm{i}/4}\right)\right]$
$\mathrm{bei}_\nu(x)$	seconda funzione di Kelvin di prima specie	$\mathrm{bei}_\nu(x) = \mathbf{Im}\left[J_\nu\left(x\, \mathrm{e}^{3\pi\,\mathrm{i}/4}\right)\right]$
$\mathrm{ker}_\nu(x)$	prima funzione di Kelvin di seconda specie	$\mathrm{ker}_\nu(x) = \mathbf{Re}\left[K_\nu\left(x\, \mathrm{e}^{\pi\,\mathrm{i}/4}\right)\right]$
$\mathrm{kei}_\nu(x)$	seconda funzione di Kelvin di seconda specie	$\mathrm{kei}_\nu(x) = \mathbf{Im}\left[K_\nu\left(x\, \mathrm{e}^{\pi\,\mathrm{i}/4}\right)\right]$
$M(a,\, b,\, z)$	funzione ipergeometrica confluente	Funzione di Kummer di prima specie
$U(a,\, b,\, z)$	funzione ipergeometrica confluente	Funzione di Kummer di seconda specie
$F(a,\, b; c; z)$	funzione ipergeometrica	L'espressione generale è $F(a,\, b; c; z) = \dfrac{\Gamma(c)}{\Gamma(b)\Gamma(c-b)} \cdot \displaystyle\int_0^1 t^{b-1}(1-t)^{c-b-1}(1-tz)^{-a}\,\mathrm{d}t$
$P_n(z)$	polinomio di Legendre	Detto anche funzione sferica
$T_n(z)$	polinomio di Chebyshev di prima specie	
$U_n(z)$	polinomio di Chebyshev di seconda specie	
$C_n^{(\alpha)}(z)$	polinomio di Gegenbauer	Detto anche polinomio ultrasferico
$P_n^{(\alpha,\,\beta)}(z)$	polinomio di Jacobi	L'intervallo di ortogonalità è $-1 \div +1$
$G(p,\, q,\, z)$	polinomio di Jacobi	L'intervallo di ortogonalità è $0 \div +1$

continua

Continua tabella 24.8

Simbolo	Significato	Note
$H_n(z)$	polinomio di Hermite	
$L_n(z)$	polinomio di Laguerre	
$L_n^{(\alpha)}(z)$	polinomio di Laguerre generalizzato	
$B_n(z)$	polinomio di Bernoulli	
$E_n(z)$	polinomio di Eulero	
$P_\nu^\mu(z)$	funzione ultrasferica di prima specie	Quando $\mu = 0$ si omette di scriverne il valore, perché la funzione coincide con il polinomio di Legendre
$Q_\nu^\mu(z)$	funzione ultrasferica di seconda specie	
$\lim_{x\to a} f(x)$	limite	
Δx	incremento finito	
δx	incremento virtuale	
$\mathrm{d}x$	differenziale	Come gli altri operatori va scritto in tondo, ma non richiede lo spazio a destra
$f(x)\Big\vert_a^b$	incremento	Cioè $f(b) - f(a)$
$\dfrac{\mathrm{d}y}{\mathrm{d}x}$	derivata	
$\dfrac{\partial y}{\partial x}$	derivata parziale	
$\dfrac{\mathrm{d}^n y}{\mathrm{d}x^n}$	derivata n-esima	
$\dfrac{\partial^n y}{\partial x^n}$	derivata parziale n-esima	L'ordine di derivazione nelle derivate parziali miste, quando non sia indifferente, è il seguente $\dfrac{\partial^2 z}{\partial x \partial y} = \dfrac{\partial}{\partial x}\left(\dfrac{\partial z}{\partial y}\right)$
$\int f(x)\,\mathrm{d}x$	integrale indefinito	

continua

Continua tabella 24.8

Simbolo	Significato	Note
$\int_a^b f(x)\,\mathrm{d}x$	integrale definito	
$\int_D f(\mathsf{P})\,\mathrm{d}D$	integrale esteso ad un dominio	Il particolare dominio D va specificato. Il punto P appartiene al dominio
$⨍_a^b f(z)\,\mathrm{d}z$	integrale principale di Cauchy	La funzione $f(z)$ è discontinua lungo la linea che congiunge a e b, e l'integrale viene calcolato come limite simmetrico a cavallo della discontinuità
$\oint_\ell f(z)\,\mathrm{d}z$	circuitazione di f lungo la linea chiusa ℓ	
$\mathcal{L}[f(t)]$	trasformata di Laplace	$F(s) = \mathcal{L}[f(t)]$
$\mathcal{L}^{-1}[F(s)]$	antitrasformata di Laplace	$f(t) = \mathcal{L}^{-1}[F(s)]$
$\mathcal{F}[f(t)]$	trasformata di Fourier	$F(\omega) = \mathcal{F}[f(t)]$
$\mathcal{F}^{-1}[F(\omega)]$	antitrasformata di Fourier	$f(t) = \mathcal{F}^{-1}[F(\omega)]$
$\widehat{\mathsf{ABC}}$	angolo	Il vertice corrisponde al punto B
$\overset{\frown}{\mathsf{AB}}$	arco	
$\overline{\mathsf{AB}}$	segmento	
$\vec{V}$	vettore	
$\lvert\vec{V}\rvert\,, V$	modulo di vettore	
$\vec{V}_1 \cdot \vec{V}_2$	prodotto scalare	Non usare mai l'operatore $\times$ per il prodotto scalare
$\begin{cases}\vec{V}_1 \wedge \vec{V}_2 \\ \vec{V}_1 \times \vec{V}_2\end{cases}$	prodotto vettore	
$\int_\ell \vec{V} \cdot \overrightarrow{\mathrm{d}\ell}$	'lavoro' di $\vec{V}$ lungo la linea ℓ	

continua

Continua tabella 24.8

Simbolo	Significato	Note
$\int_S \vec{V} \cdot \overrightarrow{\mathrm{d}S}$	flusso di $\vec{V}$ attraverso la superficie S	
$\operatorname{grad}\Phi$, $\nabla\Phi$	gradiente	Si può indicare anche con $\overrightarrow{\operatorname{grad}\Phi}$ oppure $\overrightarrow{\nabla\Phi}$
$\operatorname{div}\vec{V}$, $\nabla\cdot\vec{V}$	divergenza	
$\operatorname{rot}\vec{V}$, $\nabla\times\vec{V}$	rotore	Si può indicare anche con $\overrightarrow{\operatorname{rot}\vec{V}}$ oppure $\overrightarrow{\nabla\times\vec{V}}$
$\nabla^2\Phi$	laplaciano di uno scalare	
$\nabla^2\vec{V}$	laplaciano di un vettore	Vettore le cui componenti sono ordinatamente i laplaciani delle componenti di $\vec{V}$. Si può indicare anche con $\overrightarrow{\nabla^2\vec{V}}$
x	valore istantaneo	
X	valore efficace	Il concetto ha senso solo se $x(t)$ è periodica
$\hat{x}$, $x_{\max}$	valore massimo	Il valore massimo della funzione $x(t)$ in un intervallo prefissato $t_{\min} \div t_{\max}$
$\check{x}$, $x_{\min}$	valore minimo	Il valore minimo della funzione $x(t)$ in un intervallo prefissato $t_{\min} \div t_{\max}$
$\bar{x}$	valore medio	Il valore medio della funzione $x(t)$ in un intervallo prefissato $t_{\min} \div t_{\max}$
$\mathbf{Re}\, z$	parte reale	
$\mathbf{Im}\, z$	parte immaginaria	Se $z = x + \mathrm{i}\, y$ allora $\mathbf{Im}\, z$ dovrebbe essere uguale a y e non a $\mathrm{i}\, y$, ma talvolta è usato per $\mathrm{i}\, y$
$\lvert z\rvert$	modulo	
$\arg z$	argomento o anomalia	$z = \lvert z\rvert\, \mathrm{e}^{\mathrm{i}\arg z}$

continua

Continua tabella 24.8

Simbolo	Significato	Note
z^*	coniugato	Nei testi matematici è più comune $\bar{z}$
$f_*(s)$	paraconiugato	$f_*(s) = f(-s)$, ma se $f(s)$ è hermitiana, $f_*(\mathrm{i}\,y) = f^*(\mathrm{i}\,y)$
$\mathbf{A}$	insieme	$\mathbf{A} = \{a_1, a_2, \dots\}$. Nello stesso modo, per indicare altri insiemi, si possono usare altre lettere maiuscole, che non siano già associate ad insiemi particolari
$\emptyset$	insieme vuoto	
Ω	universo	
$\mathbf{N}$, $\mathbb{N}$	insieme dei numeri interi positivi	
$\mathbf{Z}$, $\mathbb{Z}$	insieme dei numeri interi relativi	
$\mathbf{Q}$, $\mathbb{Q}$	insieme dei numeri razionali	
$\mathbf{R}$, $\mathbb{R}$	insieme dei numeri reali	
$\mathbf{C}$, $\mathbb{C}$	insieme dei numeri complessi	
$\mathbf{A} \times \mathbf{B}$	prodotto cartesiano di insiemi	Ogni elemento del prodotto cartesiano è formato dall'accoppiamento di un elemento dell'insieme $\mathbf{A}$ con un elemento dell'insieme $\mathbf{B}$
$\mathbf{R}^n$, $\mathbb{R}^n$	insieme delle n-uple reali	Indica anche lo spazio reale a n dimensioni
$\mathbf{C}^n$, $\mathbb{C}^n$	insieme delle n-uple complesse	Indica anche lo spazio complesso ad n dimensioni
$\exists$	esiste	Esiste ed è unico: $\exists!$
$\nexists$	non esiste	
$a \in \mathbf{A}$	appartiene	

continua

Continua tabella 24.8

Simbolo	Significato	Note
$a \notin \mathbf{A}$	non appartiene	
$\mathbf{A} \ni a$	contiene	
$\mathbf{A} \not\ni a$	non contiene	
$\mathbf{A} \cap \mathbf{B}$	intersezione	
$\mathbf{A} \cup \mathbf{B}$	unione	
$\mathbf{A} \backslash \mathbf{B}$	differenza	L'insieme $\mathbf{A} \backslash \mathbf{B}$ è formato dagli elementi di $\mathbf{A}$ esclusi quelli che appartengono anche a $\mathbf{B}$
$\complement_{\Omega}\mathbf{A}$	complemento	$\complement_{\Omega}\mathbf{A} = \Omega \backslash \mathbf{A}$
$\mathbf{A} \subset \mathbf{B}$	è contenuto	$\mathbf{A}$ è un sottoinsieme di $\mathbf{B}$
$\mathbf{A} \not\subset \mathbf{B}$	non è contenuto	
$\mathbf{A} \subseteq \mathbf{B}$	è contenuto o coincide	
$\mathbf{A} \not\subseteq \mathbf{B}$	non è contenuto né coincide	
$\mathbf{B} \supset \mathbf{A}$	contiene	$\mathbf{B}$ contiene l'insieme $\mathbf{A}$
$\mathbf{B} \not\supset \mathbf{A}$	non contiene	
$\mathbf{B} \supseteq \mathbf{A}$	contiene o coincide	
$\mathbf{B} \not\supseteq \mathbf{A}$	non contiene né coincide	
$\boldsymbol{A}$	matrice	Quando la matrice ha una sola colonna (riga) si è soliti usare lettere minuscole. La matrice può essere esplicitata in uno dei modi seguenti $\boldsymbol{A} = \begin{pmatrix} a_{11} & \dots & a_{1r} \\ \vdots & \ddots & \vdots \\ a_{n1} & \dots & a_{nr} \end{pmatrix}$ oppure $\boldsymbol{A} = \begin{bmatrix} a_{11} & \dots & a_{1r} \\ \vdots & \ddots & \vdots \\ a_{n1} & \dots & a_{nr} \end{bmatrix}$

continua

Continua tabella 24.8

Simbolo	Significato	Note
$\lvert\boldsymbol{A}\rvert$, $\det \boldsymbol{A}$	determinante	La matrice $\boldsymbol{A}$ di cui si calcola il determinante deve essere quadrata. È $\det \boldsymbol{A} = \begin{vmatrix} a_{11} & \dots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \dots & a_{nn} \end{vmatrix}$
$\lVert\boldsymbol{A}\rVert$	norma	
$\operatorname{tr} \boldsymbol{A}$	traccia	$\operatorname{tr} \boldsymbol{A} = \sum_{i=1}^{n} a_{ii}$
$\boldsymbol{A}^*$	matrice coniugata	Nei testi di matematica è più frequente $\bar{\boldsymbol{A}}$
$\tilde{\boldsymbol{A}}$, ${}^t\boldsymbol{A}$	matrice trasposta	Si indica anche con $\boldsymbol{A}^{\mathrm{T}}$. Questo secondo simbolo è più adatto quando la matrice è esplicitata mediante i suoi elementi
$\bar{\boldsymbol{A}}$	matrice associata	$\bar{\boldsymbol{A}} = \tilde{\boldsymbol{A}}^*$
$\boldsymbol{A}_*$	matrice paraconiugata	$\boldsymbol{A}_*(s) = \tilde{\boldsymbol{A}}(-s)$, ma se gli elementi $a_{ij}(s)$ di $\boldsymbol{A}$ sono hermitiani $\forall\, i, j$, è $\boldsymbol{A}_*(\mathrm{i}\, y) = \bar{\boldsymbol{A}}(\mathrm{i}\, y)$
$\boldsymbol{D}$	matrice diagonale	$\boldsymbol{D} = \boldsymbol{diag}\,(a_1, \dots, a_n)$. Può essere usata qualunque altra lettera, purché ne sia definito il significato
$\boldsymbol{I}$, $\boldsymbol{1}$	matrice identità	Il secondo simbolo può essere usato solo quando il primo possa ingenerare confusione con altre grandezze
$\boldsymbol{\mathcal{G}}$	diadica	
$s_{\overline{n}i}$	capitalizzazione	Coefficiente di capitalizzazione di n annualità posticipate all'interesse i
$\ddot{s}_{\overline{n}i}$	capitalizzazione	Coefficiente di capitalizzazione di n annualità anticipate all'interesse i

continua

Continua tabella 24.8

Simbolo	Significato	Note
$a_{\overline{n}\vert i}$	attualizzazione	Coefficiente di attualizzazione di n annualità posticipate all'interesse i
$\ddot{a}_{\overline{n}\vert i}$	attualizzazione	Coefficiente di attualizzazione di n annualità anticipate all'interesse i
$\sigma_{\overline{n}\vert i}$	reintegrazione	Coefficiente di reintegrazione mediante n annualità posticipate all'interesse i
$\ddot{\sigma}_{\overline{n}\vert i}$	reintegrazione	Coefficiente di reintegrazione mediante n annualità anticipate all'interesse i
$\alpha_{\overline{n}\vert i}$	ammortamento	Coefficiente di ammortamento mediante n annualità posticipate all'interesse i
$\ddot{\alpha}_{\overline{n}\vert i}$	ammortamento	Coefficiente di ammortamento mediante n annualità anticipate all'interesse i

Tabella 24.9: Nomenclatura, simboli e unità di misura

Grandezza	**Simbolo**	**Unità SI**
angolo piano	α, β, γ, ...	rad
angolo solido	ω, Ω	sr
lunghezza	l	m
larghezza	b	m
altezza	h	m
raggio	r	m
spessore	d, δ	m
diametro	d	m
percorso curvilineo	s	m
superficie, area	S, A	m^2
volume	V, v	m^3
lunghezza d'onda	λ	m, (m/onda)
numero d'onda ($1/\lambda$)	σ	m^{-1}, (onde/m)
ondulanza ($2\pi/\lambda$)	k	m^{-1}
attenuazione spaziale	α	m^{-1}, (Np/m)
costante di fase	β	m^{-1}
costante di propagazione ($\alpha + i\,\beta$)	γ	m^{-1}
tempo	t	s
periodo	T	s, (s/ciclo)
frequenza	f	Hz, (cicli/s)
pulsazione	ω	s^{-1}
tempo di rilassamento o costante di tempo	τ	s, (s/Np)
coefficiente di smorzamento	δ	s^{-1}, (Np/s)

continua

Continua tabella 24.9

Grandezza	**Simbolo**	**Unità SI**
decremento logaritmico (T/τ)	Λ	(Np/ciclo)
velocità	v, u	m/s
velocità angolare	ω	rad/s
accelerazione	a	$\mathrm{m/s^2}$
accelerazione angolare	α	$\mathrm{rad/s^2}$
accelerazione di gravità	g	$\mathrm{m/s^2}$
costante di gravitazione	G	$\mathrm{N\,m^2/kg^2}$
velocità della luce nel vuoto	c_0	m/s
massa	m	kg
massa volumica	ϱ	$\mathrm{kg/m^3}$
densità relativa (all'acqua)	d	–
volume massico ($1/\varrho$)	v	$\mathrm{m^3/kg}$
quantità di moto	p	kg m/s
momento della quantità di moto	L	$\mathrm{kg\,m^2/s}$
momento quadratico di superficie	I	$\mathrm{m^4}$
momento di inerzia	J	$\mathrm{kg\,m^2}$
forza	F	N
coppia	T, M	N m, (N m/rad)
momento di una forza	M	N m, (N m/rad)
pressione	p	Pa
tensione normale	σ	Pa
tensione di taglio	τ	Pa
allungamento relativo	ε	–

continua

Continua tabella 24.9

Grandezza	Simbolo	Unità SI
modulo di elasticità	E	Pa
angolo di torsione	γ	rad
modulo di torsione	G	Pa
dilatazione volumica relativa	ϑ	–
modulo di compressione	K	Pa
rapporto di Poisson	μ	–
viscosità dinamica	η	Pa s
viscosità cinematica (η/ϱ)	ν	m^2/s
coefficiente di attrito	μ	–
tensione superficiale	γ, σ	N/m
energia	E	J
energia potenziale	E_p, V, Φ	J
energia cinetica	E_k, T, K	J
lavoro	W	J
potenza	P	W
rendimento	η	–
velocità del suono	c	m/s
velocità longitudinale	c_l	m/s
velocità trasversale	c_t	m/s
velocità di gruppo	c_g	m/s
flusso energetico (acustico)	P	W/m^2
fattore di riflessione (acustica)	ϱ	–
fattore di assorbimento (acustico)	α_a, α	–

continua

Continua tabella 24.9

Grandezza	Simbolo	Unità SI
fattore di trasmissione (acustica)	τ	–
fattore di dissipazione (acustica)	δ	–
livello sonoro	L_{N}, Λ	dB
corrente elettrica	i, I	A
densità di corrente	j, J	$\mathrm{A/m^2}$
carica elettrica	Q	C
densità volumica di carica	ϱ	$\mathrm{C/m^3}$
densità superficiale di carica	σ	$\mathrm{C/m^2}$
potenziale elettrico	V	V
tensione (elettrica)	V	V
impulso di tensione	U	V s
forza elettromotrice	E	V
campo elettrico	E, K	V/m
spostamento elettrico	D	$\mathrm{C/m^2}$
flusso elettrico	Ψ	C
capacità	C	F
permittività (o permettività)	ε	F/m
permittività del vuoto	ε_0	F/m
permittività relativa	ε_{r}	–
polarizzazione dielettrica	P	$\mathrm{C/m^2}$
suscettività elettrica ($\varepsilon_{\mathrm{r}} - 1$)	χ_{e}	–
elettrizzazione ($D/\varepsilon_0 - E$)	E_{i}, K_{i}	V/m
polarizzazione ($D - \varepsilon_0 E$)	P	$\mathrm{C/m^2}$

continua

Continua tabella 24.9

Grandezza	Simbolo	Unità SI
momento di dipolo (elettrico)	p	C m
campo magnetico	H	A/m
potenziale magnetico	U_m	A
forza magnetomotrice	F_m	A
induzione magnetica	B	T
flusso di induzione (magnetica)	Φ	Wb
permeabilità	μ	H/m
permeabilità del vuoto	μ_0	H/m
permeabilità relativa	μ_r	–
magnetizzazione	M	A/m
suscettività magnetica ($\mu_\mathrm{r} - 1$)	χ_m, κ	–
momento elettromagnetico	m, μ	A m^2
polarizzazione magnetica	J	T
resistenza	R	Ω
reattanza	X	Ω
impedenza	Z	Ω
fattore di qualità	Q_L, Q_C, ...	–
coefficiente di risonanza	Q	–
conduttanza	G	S
suscettanza	B	S
ammettenza	Y	S
resistività	ϱ	Ω m
conducibilità	σ, γ	S/m

continua

Continua tabella 24.9

Grandezza	Simbolo	Unità SI
induttanza (propria)	L	H
induttanza mutua	M	H
coefficiente di accoppiamento ($M/\sqrt{L_\mathrm{p}L_\mathrm{s}}$)	k	–
coefficiente di dispersione ($1-k^2$)	σ	–
riluttanza	$\mathcal{R}$, R	H^{-1}
permeanza	Λ	H
potenza reattiva	Q	V A
potenza apparente	P	V A
sfasamento	φ	rad
numero delle fasi	m	–
angolo di perdita	δ	rad
numero di spire	N, n	–
densità volumica di energia elettromagnetica	w	$\mathrm{J/m^3}$
vettore di Poynting	S	$\mathrm{W/m^2}$
potenziale vettore magnetico	$\mathcal{A}$, A	Wb/m
temperatura termodinamica	T	K
temperatura (Celsius)	t	°C
quantità di calore	Q	J
entropia	S	J/K
energia interna	U	J
energia libera ($U-TS$)	F	J
entalpia	H	J
entalpia libera	G	J

continua

Continua tabella 24.9

Grandezza	Simbolo	Unità SI
coefficiente di pressione $(\partial(\ln p)/\partial T\|_V)$	β	K^{-1}
compressibilità $(-\partial(\ln V)/\partial p\|_T)$	κ	m^2/N
coefficiente di dilatazione lineare	α	K^{-1}
coefficiente di dilatazione volumica	γ	K^{-1}
conducibilità termica	λ	$\mathrm{W}/(\mathrm{m\,K})$
calore massico	c_p, c_v	$\mathrm{J}/(\mathrm{kg\,K})$
capacità termica	C_p, C_v	J/K
rapporto dei calori massici	κ	–
flusso termico	Φ	W
flusso di calore areico	q	W/m^2
coefficiente di trasmissione termica	τ	$\mathrm{W}/(\mathrm{m}^2\mathrm{K})$
coefficiente di diffusione termica	a	m^2/s
potenza raggiante	Q, W	W
intensità energetica	I	W/sr
irradiamento	E	W/m^2
radianza	L	$\mathrm{W}/(\mathrm{m}^2\,\mathrm{sr})$
intensità luminosa	I	cd
flusso luminoso	Φ	lm
quantità di luce	Q	$\mathrm{lm\,s}$
luminanza	L	cd/m^2
illuminamento	E	lx
fattore di assorbimento	α	–
fattore di riflessione	ϱ	–

continua

Continua tabella 24.9

Grandezza	Simbolo	Unità SI
fattore di trasmissione	τ	–
indice di rifrazione	n	–
distanza di due piani reticolari adiacenti	d	m
angolo di Bragg	ϑ	rad
massa efficace dell'elettrone	m^*, m_{eff}	kg
energia di Fermi	E_{F}	J
vettore d'onda di Fermi	k_{F}	m^{-1}, (rad/m)
coefficiente di Seebeck	S	V/K
coefficiente di Peltier	Π	V
coefficiente piezoelettrico (polarizzazione/sforzo)	d_{mn}	C/N
temperatura caratteristica di Weiss	Θ_{W}	K
temperatura di Curie	T_{C}	K
temperatura di Neel	T_{N}	K
coefficiente di Hall	R_{H}	$\mathrm{V\,m^2/A^2}$
quantità di sostanza	n	mol
massa molare	M	kg/mol
volume molare	V_{m}	$\mathrm{m^3/mol}$
energia interna molare	U_{m}	J/mol
capacità termica molare	$C_{p\mathrm{m}}$, $C_{v\mathrm{m}}$	J/(mol K)
entropia molare	S_{m}	J/(mol K)
concentrazione di un costituente	c	$\mathrm{mol/m^3}$
molalità di un soluto	m	mol/kg

continua

Continua tabella 24.9

Grandezza	**Simbolo**	**Unità SI**
dose assorbita	D	Gy
energia impartita massica	z	Gy
indice di dose assorbita	D_{i}	Gy
rateo di dose assorbita	$\dot{D}$	Gy/s
equivalente di dose	H	Sv
esposizione	X	C/kg
rateo di esposizione	$\dot{X}$	C/(kg s)
attività di un radio nuclide	A	Bq

Capitolo 25

Divisione in sillabe

L'algoritmo per dividere in sillabe le parole in fin di riga è una specialità del sistema TEX, ed è stata incorporata in diversi altri programmi, tra i quali merita segnalare OpenOffice e NeoOffice come esempi di software Open Source, a cui possono collaborare tutti gli utenti che ne hanno la possibilità e la competenza; un po' come succede per il sistema TEX.

Oltre a quei due word processor usano questo stesso metodo anche LibreOffice, Apache OpenOffice, Inkskape, GIMP, Scribus, InDesign, Illustrator. Lo usano anche il client JavaScript con il modulo *hyphenator.js*, e i browser Web e i client di posta elettronica Gecko di Mozilla (Firefox per calcolatori normali e per dispositivi mobili); WebKit di Apple e Adobe (Safari, Konqueror); Blink di Google (Chromium e Chrome, Opera, Web Browser per Android). La lettura di questo capitolo permette quindi di capire il comportamento di molti programmi di uso comune, che non hanno legami con il sistema TEX, ma che hanno importato da questo il suo algoritmo di divisione in sillabe.

25.1 Quando viene eseguita la cesura

TEX attiva l'algoritmo di sillabazione quando, terminata la lettura di un capoverso dal file sorgente e dopo aver trasformato le informazioni contenute nel file sorgente in una lunga linea di caratteri e altre informazioni specifiche con l'aiuto dei pacchetti *inputenc* e *fontenc*, deve accingersi a spezzare questa lunga riga nelle varie righe giustificate che appariranno nel file di uscita.

Si noti bene: tutte le macro presenti nel file sorgente sono state già elaborate al punto che la lunga riga contiene solo caratteri e alcuni comandi 'primitivi' che non producono testo e che il motore di composizione usa solo nel momento di creare la pagina di uscita.

Per quel che riguarda la sillabazione si noti che il file di ingresso poteva contenere caratteri codificati in diversi modi, mentre il file di uscita, e quindi la lunga riga, contiene solo caratteri codificati secondo le prescrizioni dei font usati

per l'uscita. Se si usa il vecchio encoding OT1 per i font di uscita, qualunque lettera con segni diacritici non è compresa nella polizza dei caratteri e deve venire ottenuta componendo il carattere di base con il segno diacritico che gli compete; questa composizione si ottiene per sovrapposizione; il comando per eseguire questa sovrapposizione è quindi un comando primitivo ancora presente nella stringa che costituisce la lunga riga da spezzare. Se invece si usa la codifica T1, le lettere con diacritici delle lingue occidentali sono già tutte presenti nella polizza del font usato e quindi la lunga riga contiene solo lettere (sempre trascurando le altre informazioni codificate che verranno usate solo nel processo di formazione della pagina di uscita, alcune delle quali solo durante la scrittura nel file di uscita di una pagina completa).

Ciò premesso, bisogna capire come fa TEX a individuare una parola, ciò che il programma crede che sia una parola; per TEX una parola è una sequenza di lettere maiuscole o minuscole, tratte dallo stesso font, che comincia dopo uno spazio o può essere preceduta da segni di interpunzione in senso lato (per esempio: parentesi, virgolette, ecc.) e finisce con il primo token che non abbia le caratteristiche di una lettera. Questa stringa non può essere seguita da token diversi da una lettera che appartengano a certe categorie, per esempio non può essere seguita dal comando `\footnote`. Non può nemmeno essere preceduta da una macro, perché lo spazio che separa la macro dalla stringa di lettere serve già all'interprete TEX per capire che la macro è terminata; la stringa può però costituire l'argomento di una macro. Non deve nemmeno essere la prima parola dopo un cambiamento di modo, per esempio da verticale a orizzontale; questo significa che la prima parola di un capoverso non viene divisa in sillabe.

Lavorando con `pdflatex` il lettore può sfruttare il comando `\showhyphens`[1] per vedere come TEX divida in sillabe; usando *`babel`* con l'opzione per l'italiano potrebbe vedere che cosa succede scrivendo;

```
\showhyphens{macroistruzione macro"istruzione}
```

e troverebbe sullo schermo o nel file log un messaggio del tipo

```
Underfull \hbox (badness 10000) in paragraph at lines 5300--5300
[] \fontiii ma-croi-stru-zio-ne ma-cro-istru-zio-ne
```

A seconda del sistema operativo e della particolare distribuzione del sistema TEX, il nome `\fontiii` potrebbe essere sostituito da qualcosa d'altro; in ogni caso esso rappresenta la descrizione interna del font in uso nel momento in cui viene eseguito il comando `\showhyphens`.

L'annotazione 'Underfull \hbox (badness 10000) in paragraph at lines ...' può venire ignorata; questo è il messaggio che TEX emette ogni volta che deve allargare troppo lo spazio interparola per giustificare una riga.

[1] Il comando `\showhyphens` non funziona bene con il programma di composizione `xelatex`. Qualunque sia il programma di composizione il lettore può caricare il pacchetto *`testhyphens`* e controllare la sillabazione delle parole inserendole dentro l'ambiente *`checkhyphens`*; le parole divise in sillabe non appaiono nel file `.log` ma direttamente nel documento.

La parte più interessante viene dopo; infatti 'ma-croi-stru-zio-ne' è la divisione in sillabe che TEX eseguirebbe spontaneamente, senza sapere che si tratta di una parola composta, mentre 'ma-cro-istru-zio-ne' è la divisione in sillabe conseguente all'uso del carattere attivo `"`, che, come si è già detto, serve appunto (tra l'altro) per indicare una divisione etimologica, piuttosto che fonetica.

Si noti che la potenziale parola talvolta non coincide con quello che gli umani sanno essere una parola della loro lingua o di un'altra lingua straniera.

Se si usasse la codifica OT1, invece della codifica T1, gli accenti vengono collocati sulle vocali (o sopra o sotto altre lettere, come per esempio: ç oppure č in molte altre lingue diverse dall'italiano) mediante un comando primitivo che si chiama `\accent`; quindi anche se uno scrive `qualità`, quando TEX interpreta il contenuto del file sorgente, sostituisce la 'à' con una sequenza di istruzioni, in particolare sostituisce la stringa iniziale con `qualit\accent18a`; in questo modo per TEX quello che lui crede essere una parola è la stringa 'qualit'; all'interno di questa stringa esso trova un solo punto di divisione, 'qua-lit' e quindi l'effetto complessivo sarebbe di dividere la parola in 'qua-lità' invece che in 'qua-li-tà'.

Queste cesure mancate potrebbero essere talvolta causa di composizioni in cui qualche riga sporge fuori dalla giustezza, oppure qualche riga è stata spaziata troppo per consentire la giustificazione; tuttavia il problema è facilmente risolubile; basta usare la direttiva

```
\usepackage[T1]{fontenc}
```

come si è già detto diverse volte.

Quando si esamina il file `.log` per cercare di capire perché qualche mancata cesura in fin di riga abbia provocato righe troppo lunghe o troppo spaziate, si scopre sempre che le cause sono quelle descritte sopra: una macro precede troppo da vicino o segue troppo da vicino una stringa di lettere che potrebbe essere una parola del linguaggio; oppure si è usato un font che non consente la cesura, come per esempio un font della famiglia a spaziatura fissa.

Un espediente elementare sarebbe quello di dividere a mano: basta inserire il comando `\-` all'interno di una parola nel punto in cui si vorrebbe eseguire la cesura; in questo testo non è mai stato usato.

Ma più in generale basta inserire una spaziatura orizzontale di ampiezza nulla fra la parola e l'oggetto 'troppo vicino', come per esempio una macro o una nota o all'inizio di un capoverso, o prima della prima parola che segue un comando `\item` in una lista. Ciò si ottiene usando comandi di basso livello come per esempio

```
\hskip 0pt
```

dove `\hskip` è il comando primitivo che inserisce lo spazio orizzontale specificato immediatamente dopo, nel nostro caso 0 pt.

In certi casi questa operazione diventa risolutiva; faccio solo un esempio: la macro troppo vicina potrebbe essere una dichiarazione di cambiamento di font; il meccanismo di sillabazione non funziona, quindi, per la prima parola

dopo un cambiamento di font; raramente questo è un vero problema, ma si presenta più frequentemente di quanto non si creda. Un comando che implica un cambiamento di font è `\emph` che cambia il font corrente in uno che non abbia la stessa inclinazione, cioè da tondo passa a corsivo e viceversa; da inclinato passa a tondo, e via di questo passo. Se si dovesse scrivere "nell'*eventualità*", nel file sorgente si inserisce `nell'\emph{eventualità}`. Fin qui è scontato e tutto dovrebbe svolgersi normalmente; ma se questa breve locuzione capita verso la fine della riga, non può venire divisa in sillabe, come si vede bene dalla *minipage* riquadrata di sinistra:

Bisogna osservare che nell'*eventualità*

Bisogna osservare che nell'*eventualità*

Si osserva infatti che l'ultima locuzione non è divisa in sillabe e che per mantenere la composizione giustificata pdflatex ha inserito spazi inaccettabili nelle righe precedenti. Però nella *minibox* riquadrata di destra è stato aggiunta la specificazione `\hskip 0pt` prima della parola 'eventualità', quando a rigore il cambiamento di font avviene prima che il programma prenda in considerazione quella parola e quindi la sillabazione avviene correttamente. Nel primo caso invece il programma considerava come "parola" tutta la stringa `nell'\emph{eventualità}` all'interno della quale avveniva un cambiamento di font.

Ora il comando `\emph` richiama al suo interno la dichiarazione `\em` che presiede al cambiamento di font; se noi modifichiamo questa macro aggiungendole in coda il grumo di colla di larghezza nulla che separa quindi la dichiarazione dalla parola da sillabare, potremmo risolvere il problema; infatti usiamo uno dei comandi del pacchetto *etoolbox* già descritti in parte nel paragrafo 29.8.4.2. Usiamo il comando (robusto) `\addto` che permette di aggiungere in coda altro codice alla definizione di una macro; la sintassi è la seguente"

`\addto`⟨*macro*⟩`{`⟨*codice aggiuntivo*⟩`}`

Noi pertanto nel preambolo o in un file di macro personali scriviamo:

```
\makeatletter % Necessario solo per il preambolo
\addto\em{\nobreak\hskip\z@}
```

dove `\z@` rappresenta uno spazio di `0pt`; il comando `\nobreak`, quando si parla di comandi che implicano un cambiamento di font, è quanto mai opportuno se prima del comando, come nel nostro caso, non c'è nessuno spazio, anzi è vietato andare a capo dopo l'apostrofo; dopo di che l'esempio precedente diventa:

Bisogna osservare che nell'*eventualità*

Come si vede ora la sillabazione funziona correttamente. Definendo un breve comando come `\hz`:

```
\makeatletter % Necessario solo per il preambolo
\newcommand*\hz{\nobreak\hskip\z@}
```

in un file di macro personali o nel preambolo, si può anche inserire a mano questo breve comando in ogni posizione in cui sia necessario un grumetto di gomma di larghezza nulla per separare ciò che precede troppo da vicino una parola impedendone la sillabazione. In modo analogo si può operare (ma solo scrivendo in italiano) mediante l'uso del carattere attivo `"`, come descritto nel prossimo paragrafo.

È bene conoscere questo meccanismo al fine di porre rimedio alle rare circostanze in cui il meccanismo di sillabazione non svolga il suo compito come ci si aspetterebbe.

25.2 La sillabazione fonetica oppure etimologica

Vale la pena di aggiungere una osservazione. Nessun meccanismo automatico è infallibile; la sillabazione in italiano funziona molto bene, ma non si può escludere che un neologismo particolare o una parola tecnica possa risultare divisa in sillabe in modo scorretto, specialmente se è ricavato da una radice straniera. Per esempio la parola tecnica dell'ambito medico 'dispepsia" viene divisa in 'di-spep-sia'; ovviamente questo è un caso semplice da trattare mediante l'uso del carattere attivo `"`, perché scrivendo `dis"pepsia`, la parola risulta divisa in modo etimologico. Dalla chimica si potrebbe prendere un lunga parola composta, per esempio *desclorfeniraminacloridrato* che può venire scritta nella forma `des"clor"fenir"amina"clor"idrato` che porta sempre alla divisione in sillabe corretta, anche nei punti non esplicitamente indicati: *des-clor-fe-nir-ami-na-clor-idra-to*. Attenzione: perché il moncone di parola *-ami-* non viene diviso in *-a-mi-*? Perché il segno `"` spezza la stringa che TEX crede essere una parola in tanti monconi che TEX tratta come parole intere, e in nessuna parola TEX stacca una sillaba iniziale o finale formata da una sola lettera; questo fatto verrà commentato meglio più avanti.

In inglese il meccanismo di sillabazione sembra funzionare correttamente in poco più del 90% dei casi. In francese l'algoritmo funzionerebbe benissimo, nonostante le numerose lettere con diacritici, ma le regole della buona educazione impediscono di dividere in sillabe parole che lascino nella riga contenente la cesura, o nella riga seguente, monconi di parole che hanno significati volgari; a causa di ciò alcune parole non vengono divise affatto o vengono divise in monconi talvolta troppo lunghi. In tedesco il problema è veramente difficile a causa delle numerosissime parole composte che vanno divise etimologicamente, talvolta con un cambio di ortografia. Tuttavia l'algoritmo viene usato con soddisfazione da tutti, e nei pochi casi in cui fallisce ci sono almeno tre strade assai semplici da seguire:

1. Si marcano a mano con `\-` i punti di divisione di quelle poche parole che nelle bozze risultano divise in modo scorretto; con l'italiano ciò è

estremamente raro, per non dire impossibile, perché le norme UNI che regolano la materia ammettono come corrette tutte le divisioni fonetiche; certo con parole straniere o derivate da radici straniere questo potrebbe non essere corretto.

2. Si usa per l'italiano il carattere attivo `"` nelle parole composte che si desidera dividere etimologicamente; si noti che le norme UNI che regolano questa questione in italiano accettano la divisione fonetica anche per le parole composte, ma in uno scritto tecnico è preferibile usare la divisione etimologica. Nelle altre lingue straniere, tranne in inglese, le opzioni di *babel* prevedono quasi sempre un comando che utilizza il carattere attivo `"`, per esempio `"-`, che consente di svolgere la stessa funzione che `"` da solo svolge in italiano; bisogna consultare la parte di documentazione di *babel* relativa alla lingua specifica. Dalla versione 3.9 di *babel* la documentazione relative alle lingue si legge semplicemente dando da terminale il comando `texdoc babel-`⟨*lingua*⟩ (per esempio `texdoc babel-italian` oppure `texdoc bebel-german`, eccetera).

3. Si ricorre alla lista di parole divise in sillabe contenute nell'argomento del comando `\hyphenation`; per l'italiano, per esempio, se si dovesse usare spesso la parola 'macroistruzione' al singolare e al plurale basterebbe scrivere nel preambolo:

   ```
   \hyphenation{ma-cro-istru-zio-ne ma-cro-istru-zio-ni}
   ```

   Nel preambolo si possono usare diverse liste introdotte mediante il comando `\hyphenation`; i loro contenuti vengono raccolti tutti assieme in una apposita memoria del programma, il quale consulta sempre questa lista prima di attivare l'algoritmo di sillabazione vero e proprio.

   Con lingue che usano la declinazione (come succede in parte per l'italiano) e la coniugazione, queste liste potrebbero diventare immense. In italiano un sostantivo richiede due voci: singolare e plurale; per un aggettivo ne possono essere richieste quattro: singolare e plurale, maschile e femminile; per un verbo occorrono una sessantina di voci. In italiano si ricorre assai raramente a questo meccanismo, ma, per esempio, per l'inglese ogni installazione del sistema TEX prevede la lettura automatica iniziale di una lista di eccezioni di diverse centinaia di parole.

25.3 Come fa TEX a dividere in sillabe

Tuttavia, superati questi punti, vediamo come fa TEX a dividere in sillabe.

Esso spezza ogni parola che si trovi verso la fine di ogni riga nelle sue componenti di una sola lettera, di due lettere, di tre lettere, eccetera, ma con un procedimento sequenziale che non comporti tutte le possibili scomposizioni; nel fare questo esso è guidato dai 'pattern di sillabazione'. Questi sono contenuti in

.d	e	l	l	’	i	s	t	r	u	z	i	o	n	e.
.d	0 e	0 l	0 l	0 ’	0 i	0 s	0 t	0 r	0 u	0 z	0 i	0 o	0 n	0 e.
		1 l		’ 2		1 s	2	1 r		1 z			1 n	
			1 l				1 t							
		2 l	0 l				t	2 r						
.d	0 e	2 l	1 l	0 ’	2 i	1 s	2 t	2 r	0 u	1 z	0 i	0 o	1 n	0 e.

Tabella 25.1: Pattern usati per dividere in sillabe *dell’istruzione*

un file che in fase di costruzione del formato vengono allocati in una struttura dati molto veloce da esplorare; è per questo che bisogna sempre essere sicuri che detto file sia stato elaborato nella costruzione del file di formato, altrimenti la sillabazione per la lingua cui corrispondono quei pattern è impossibile.

I pattern sono formati da monconi di parole in cui è indicata la possibilità di dividere in sillabe mediante delle cifre pari (non dividere) o dispari (dividi pure) che possono andare da 0 a 9. Se fra le stesse due lettere di pattern diversi compaiono cifre diverse, non si tratta di una contraddizione, ma fra le due indicazioni prevale quella con la cifra più alta. Per esempio, il pattern `1b` dice che si può dividere prima della ‘b’, ma non dopo (la cifra assente implica la cifra zero); ma il pattern `2bb` dice che si può andare a capo *dopo* la prima ‘b’, ma *non* prima. Infatti, tenendo conto dei due pattern, il secondo si legge come se fosse `2b1b0`, il che conferma quanto appena descritto a parole.

Prendiamo un esempio più complesso, per esempio la ‘parola’ con elisione e apostrofo: *dell’istruzione*. Conviene rifarsi alla tabella 25.1; in questa tabella la parola iniziale è scomposta nelle sue componenti ed è preceduta e seguita da due segni di punto; questi rappresentano per TEX i delimitatori della stringa di lettere che lui ritiene essere una parola. Nella seconda riga tutti i codici di cesura pari al valore di default ‘zero’ sono intercalati alle lettere. Nella terza e nella quarta riga sono indicati i pattern che coinvolgono una sola lettera, contornati da rispettivi codici di consenso o interdizione della cesura; come si vede questi pattern coinvolgono solo le consonanti e l’apostrofo. Nella quinta riga sono riportati i pattern che coinvolgono due lettere insieme ai rispettivi codici di consenso e interdizione della cesura; per l’italiano non esistono altri pattern di tre o più lettere che si possano trovare nella parola indicata, quindi non ci sono altre righe da scrivere nella tabellina. Perciò nell’ultima riga, oltre alle lettere componenti, sono riportate le cifre più alte che compaiono in ogni colonna di cifre. Ne consegue che le cesure possono cadere solo fra le due ‘l’, tra la ‘i’ e la ‘s’, tra la ‘u’ e la ‘z’ e tra la ‘o’ e la ‘n’, dove compaiono i codici dispari.

La tabella 25.2 mostra un esempio ancora più complesso; non lo si commenta, perché il procedimento esposto in relazione alla tabella 25.1 è il medesimo. Questa volta la parola da dividere in sillabe è *discinesia*, che i medici pronunciano con il prefisso ‘dis’ separato dalla parola ‘cinesia’. Il risultato infatti porta alla divisione etimologicamente corretta *dis-ci-ne-sia*. Il lungo pattern che compare nella quinta riga, serve appunto per specificare l’eccezione alla regola di non separare la ‘s’

.d	i	s	c	i	n	e	s	i	a.
.d 0	i 0	s 0	c 0	i 0	n 0	e 0	s 0	i 0	a.
	1 s	2		1 n		1 s	2		
		1 c							
.d 0	i 2	s 3	c 0	i 0	n 0	e			
.d 0	i 2	s 3	c 0	i 1	n 0	e 1	s 2	i 0	a.

Tabella 25.2: Pattern usati per dividere in sillabe *discinesia*

impura (che in questo caso, insieme alla 'c' e alla 'i' formerebbe un 'trigramma' che normalmente si pronuncia diversamente dalle singole lettere componenti); il pattern è sufficientemente lungo per isolare la radice di 'discinesia' e dei suoi derivati, come per esempio 'discinetico', ma non coinvolge altre parole che cominciano nello stesso modo, per esempio 'disciplina, 'discinto', eccetera.

Per l'inglese i pattern sono quasi 5000, le eccezioni introdotte con il comando `\hyphenation` sono alcune centinaia e la probabilità di una divisione errata è bassa, ma non nulla.

In italiano i pattern sono circa 330, la lista delle eccezioni è nulla e negli ultimi 10 anni non sono stati segnalati errori; tutto ciò la dice lunga sulla facilità della divisione fonetica dell'italiano e della difficoltà della divisione ritmica dell'inglese; in questa lingua due parole ortograficamente identiche ma che si pronunciano con accenti ritmici diversi, richiedono divisioni diverse: *record* (nome) e *record* (verbo) si dividono rispettivamente in *rec-ord* e *re-cord*!

I pattern per l'italiano sono tradizionalmente contenuti nel file `ithyph.tex`, ma le recenti estensioni all'uso dei font codificati secondo la codifica UNICODE ha portato alla creazione di un altro sistema di file per la definizione dei pattern per le varie lingue. Secondo questo nuovo schema, il caricamento dei pattern è suddiviso fra tre file, il primo, `loadhyph-it.tex`, è semplicemente un *loader* che esegue alcune definizioni e poi carica i file dei pattern veri e propri; questi ultimi per l'italiano si chiamano `hyph-quote-it.tex` e `hyph-it.tex`; il primo riguarda i pattern relativi all'apostrofo che in codifica ASCII è lo stesso carattere usato per la virgoletta alta semplice, mentre in codifica UNICODE si tratta di due caratteri distinti; il secondo contiene i pattern veri e propri della lingua italiana. La lista vera e propria di pattern per l'italiano non cambia dal vecchio al nuovo formato, ma per altre lingue ci possono essere diversi cambiamenti legati alla presenza di caratteri specifici di quella lingua che, se fossero scritti con la codifica interna di TEX per i numeri esadecimali, oppure fossero codificati con numeri decimali, risulterebbero del tutto incomprensibili; il *loader* allora definisce quei caratteri con delle sequenze di controllo, per esempio `\ae`, al posto di `^^^^00e6`, in modo da individuare direttamente la legatura 'æ'.

Esercizio 25.1 Il lettore cerchi nell'elenco dei pattern (nel file `hyph-it.tex`) quelli che corrispondono alle possibili divisioni di una parola a sua scelta e controlli a mano la divisione prodotta dai pattern; successivamente inserisca la stessa parola come argomento del comando `\showhyphens` e verifichi

se LaTeX divide come egli ha trovato a mano. Per semplicità usi la classe *minimal*.

Esercizio 25.2 Perché la lista dei pattern `hyph-quote-it.tex` contiene l'attribuzione di un codice strano all'apostrofo e poi la lista dei pattern contiene anche pattern con uno o due apostrofi?

Esercizio 25.3 Il file dei pattern non contiene nulla per dividere in sillabe in modo etimologico la parola *dispepsia*, anche perché il dizionario Garzanti indica solo la divisione *di-spep-sia*.

Quale pattern bisognerebbe aggiungere a quelli esistenti per dividere questa parola e i suoi derivati, come per esempio *dispeptico*, in modo etimologico?

25.4 Bruttezza residua

In alcune circostanze, pur dividendo in sillabe le parole in fin di riga, rimangono delle righe sporgenti fuori della giustezza oppure troppo 'brutte' perché lo spazio interparola è stato allargato troppo.

Non è il caso di preoccuparsi troppo se l'indice di bruttezza (*badness* in inglese) è di qualche centinaio o di poche migliaia; il messaggio di bruttezza, che viene esposto sullo schermo e trascritto nel file `.log`, comincia a diventare preoccupante se si supera il valore 5000, ma certamente la bruttezza è 'infinita' se il messaggio riporta una bruttezza di 10 000.

Per la bruttezza delle righe si può specificare nel preambolo un limite sotto il quale il programma non deve preoccuparsi; il parametro `\hbadness` può venire impostato nel preambolo ad un valore diverso a seconda del tipo di documento e del suo contenuto; per esempio

```
\hbadness=5000
```

permette di eliminare tutti i messaggi relativi alle righe la cui bruttezza sia inferiore al valore 5000.

Fin qui si sta parlando della bruttezza relativa alle righe tropo "vuote" la cui bruttezza sia inferiore al valore specificato. La bruttezza è presente anche quando le righe sono troppo piene, al punto di uscire dalla giustezza. Per ridurre i messaggi relativi a queste righe si potrebbe specificare un valore per la lunghezza `\hfuzz`, per esempio:

```
\hfuzz=1pt
```

cosicché non ci siano messaggi per righe che "forano" la giustezza di meno di 1 pt. Quando si usa il pacchetto *microtype* è difficile che questo succeda ma non è impossibile; quindi non è fuori luogo specificare un buon valore anche se è ammesso un certo grado di protrusione dei caratteri quando si usa questo pacchetto. Ricordiamo che un punto tipografico equivale a circa un terzo di millimetro; l'occhio lo vede benissimo, ma generalmente non ne è disturbato.

In modo analogo si possono eliminare alcuni messaggi per la bruttezza verticale, che TEX emette quando deve allargare troppo la gomma dei contrografismi verticali, durante la composizione della pagina; il parametro specifico si chiama `\vbadness`; talvolta la bruttezza verticale può essere notevole, ma, a parte stabilire un valore per il parametro appena menzionato, se si vuole rendere la composizione veramente professionale bisogna apportare modifiche al testo da comporre. Generalmente questa bruttezza verticale dipende dal fatto che un oggetto ingombrante (come ad esempio il titolino di una sezione seguito da almeno due righe di testo) non trova posto al fondo di una pagina. Allungando il testo che precede l'oggetto ingombrante si riduce l'ammontare dell'allargamento della gomma verticale. Analogamente al caso delle righe che sforano la giustezza, si può specificare un valore `\vfuzz` per lo sforamento dell'altezza della gabbia sotto alla quale il programma di composizione non deve emettere messaggi.

Tornando alle righe brutte si può procedere come per le pagine brutte: si modifica il testo. Tuttavia sarebbe meglio conoscere il meccanismo con cui TEX divide i capoversi in righe. Siccome però questo argomento si addentra nelle parti più tecniche, si rinvia il lettore al testo di base, il TEXbook, dove l'argomento è sviscerato nei dettagli. Una breve descrizione di questo meccanismo è riportata nel capitolo 27.

Qui il lettore si accontenti di sapere che la brutta divisione che TEX ha trovato è quella che rende minima la bruttezza dell'intero capoverso; dunque c'è poco da fare per ridurre la bruttezza, visto che questa è la minima possibile. Si può tentare, per esempio, di introdurre degli altri punti di cesura mediante il comando `\-`; ma si veda nel seguito che cosa potrebbe succedere.

25.5 I pattern per la lingua italiana

I pattern della lingua italiana si possono creare semplicemente implementando le regole grammaticali specificate in qualunque grammatica, salvo che le grammatiche generalmente si riferiscono alla lingua conosciute dai bambini delle elementari e delle medie; generalmente non prevedono l'uso di termini tecnici e/o di prole italiane basate su una radice straniera. Le regole, comunque sono queste.

1. Ogni sillaba contiene una vocale o un dittongo o un trittongo; un dittongo è formato da due vocali di cui almeno una sia una 'i' o una 'u' non tonica; un trittongo è formato da tre vocali di cui almeno una sia una 'a' oppure una 'e' oppure una 'o' e le altre due siano delle 'i' o delle 'u' non toniche. Quattro vocali di seguito contengono sempre un trittongo oppure sono due dittonghi. Le vocali di un dittongo o di un trittongo fanno parte di un'unica sillaba.

2. Ogni consonante non viene mai separata dalla vocale che la segue. Di conseguenza in ogni parola che cominci con un gruppo di consonanti, queste fanno parte della sillaba che contiene la prima vocale che le segue;

in una parola che termini con una o più consonanti, queste formano sillaba con la vocale che le precede.

3. Un gruppo di due o più consonanti in posizione intervocalica possono essere separate fra due sillabe adiacenti nel punto più a sinistra che lasci alla sua destra un gruppo di consonanti che possa trovarsi all'inizio di una parola italiana; per questo motivo le consonanti doppie si dividono fra due sillabe adiacenti, così come il gruppo 'cq', perché non si trovano mai all'inizio di una parola italiana. per questo stesso motivo la consonante 's' non si divide mai dalle consonanti (diverse da 's') che la seguono; per questo stesso motivo le consonanti liquide ('l' ed 'r') e nasali ('m' ed 'n') si staccano sempre dalle consonanti che le seguono; sempre per questo motivo non si dividono i gruppi 'gl' e 'gn'.

4. È consentita, ma non è obbligatoria, la divisione etimologica anche se viola le regole precedenti.

La regola 3 è la più discutibile, anche se può essere applicata tranquillamente scrivendo in italiano di registro semplice, come quello usato dai bambini, ma non prende in considerazione diverse le parole che possono essere present in un registro elevato.

La parola 'wagneriano' va divisa in *vag-ne-ria-no* anche se viola la regola suddetta; lo stesso vale per *watt-me-tro, mass-me-dio-lo-go* e altre simili parole composte; lo stesso vale per *tay-lo-ri-smo, new-to-nia-no, new-yor-ke-se, max-wel-lia-no, leish-ma-nio-si* e con simili parole con radice straniera. ma anche la regola delle consonanti iniziali di parole italiane non è tanto valida; come ci si composta con i gruppi di consonanti iniziali di parole italiane come *bdelio, cnidio, ctenidio, ftaleina, gmelinite, pneuma, psiche, pteridina, tmesi*? La parola 'istmo' si divide *ist-mo* o *is-tmo* visto che esiste la parola italiana 'tmesi' che comincia con 'tm'? D'accordo, sono parole italiane rare e molto tecniche, tutte di origine greca, ma come si fa ad enunciare la regola 3 per escludere le parole di origine greca? Che cosa ne sa la maggior parte delle persone dell'etimologia delle parole? Certo si può consultare un vocabolario, ma molte di quelle parole mancano da quasi tutti i vocabolari. Inoltre sono pochi i vocabolari che indicano la divisione in sillabe delle parole; fra i pochi che lo fanno il Garzanti indica la divisione *a-pne-a* rispettando la regola come enunciata nella regola 3, ma se si escludono le parole di origine greca, la divisione diventa *ap-ne-a* molto più naturale e conforme alla pronuncia dell'italiano. Ma. . . : 'apnea' è una parola composta, quindi è valida anche la divisione etimologica *a-pne-a*.

Ci sono poi altre osservazioni da fare in merito ai gruppi di vocali Infatti i pattern per l'italiano non dividono i dittonghi (ed è giusto), ma non dividono nemmeno gli iati; questa decisione è stata presa con una motivazione 'psicologica': il lettore si trova a disagio se trova un gruppo di due o più vocali spezzato fra due righe, anche quando la grammatica lo consentirebbe; si è deciso di spezzare i gruppi di tre o più vocali se queste contengono almeno un dittongo o un 'trittongo', ma lasciando un eventuale trittongo indiviso; quindi 'quieto' si divide

solo in 'quie-to' perché contiene un trittongo, ma 'maieutica' si divide in 'ma-ieu-ti-ca' perché le quattro vocali contengono un trittongo. Però 'aiuola' viene diviso solamente in 'aiuo-la' e non in 'a-iuo-la', come la grammatica consentirebbe, perché TEX non divide dopo sillabe iniziali o prima di sillabe finali 'troppo corte'.

Una sillaba iniziale o finale è troppo corta se contiene meno lettere del numero specificato dai parametri `\lefthyphenmin` e `\righthyphenmin`; per l'italiano entrambi questi parametri valgono 2, mentre per l'inglese, il francese ed altre lingue essi valgono rispettivamente 2 e 3.

Con i vincoli tipografici e psicologici indicati, pertanto, parole come 'idea', 'maestro', 'eroe', 'croato' perdono dei possibili punti di divisione grammaticale, ma rispettano le regole tipografiche e quelle psicologiche. All'occorrenza è sempre possibile dividere mediante l'uso del carattere attivo `"`, oppure mediante la predisposizione di elenchi di parole divise da dare in pasto al comando `\hyphenation`; chi scrive, però non ne ha mai avuto bisogno, specialmente da quando pdflatex può usare il pacchetto *microtype* che consente una composizione migliore e con pochissime parole divise in sillabe in fin di riga. Questo testo ha pagine intere in cui non compare nemmeno una cesura o dove compare una cesura sola.

25.6 Come rimediare a righe sporgenti dalla giustezza

Perciò in quelle rare circostanze in cui la giustezza non consente la composizione perfetta nemmeno usando *microtype*, per un dato capoverso diviso in modo brutto, si potrebbe formare un gruppo dove si impostano i due parametri `\lefthypenmin` e/o `\righthyphenmin` a valori più bassi e/o si inseriscono le divisioni esplicite, per esempio, di `pa"e"se` oppure di `i"de"a"le`. La bruttezza si sposta dalla forma del capoverso alle cesure scelte in modo grammaticalmente corretto ma psicologicamente infelice.

Un'altra possibilità è quella di usare l'ambiente *sloppypar* con le impostazioni e la sintassi seguente:

```
\begin{sloppypar}\tolerance=⟨tolleranza⟩
⟨capoverso da comporre⟩
\end{sloppypar}
```

Il valore ⟨*tolleranza*⟩ viene specificato mediante un numero inferiore a 10 000, ma altrimenti piuttosto alto; chi scrive spesso usa il valore 9999. Si tratta di una forte tolleranza, che però, secondo TEX, non è infinita. Il capoverso avrà una certa bruttezza, ma non sarà infinitamente brutto.

La soluzione migliore, se il contenuto del capoverso lo consente, è però quella di modificare il testo; spesso basta aggiungere o togliere una parola, oppure scambiare di posto due parole, oppure riformulare un periodo o una frase. Se il capoverso non è troppo corto le possibilità sono innumerevoli.

Il lettore attento avrà notato che in questo testo le cesure in fin di riga sono rarissime; in parte è merito dell'uso del pacchetto *microtype* e in parte del fatto che la giustificazione delle righe mediante il sapiente allargamento e restringimento dello spazio interparola, guidato dall'algoritmo di minimizzazione della bruttezza di ciascun capoverso, rende la divisione in sillabe piuttosto rara. Questo è uno dei pregi del programma di composizione tipografica pdftex che lo rende insuperabile in molte applicazioni.

25.7 I file di pattern

Con le ultime distribuzioni di TEX Live e di MiKTEX non sarebbe il caso di preoccuparsi delle lingue con le quali si può dividere in sillabe un testo scritto in una delle moltissime lingue che LATEX sa gestire tramite il pacchetto *babel* e i loro vari file di descrizione. L'ultima volta che ho controllato, con entrambe le distribuzioni sopra citate vengono caricati tutti i pattern per tutte le lingue che LATEX sa gestire. Oggi con la versione 2014 di TEX Live le lingue gestibili sono 80.

Con le distribuzioni di base o ridotte al minimo indispensabile, bisogna leggere la documentazione che accompagna la propria distribuzione per sapere come fare per attivare le lingue che non sono attivate in prima installazione.

Generalmente con TEX Live non c'è bisogno di preoccuparsi perché l'installazione completa è quella di default, e per eseguire un'installazione parziale bisogna mettercela tutta per specificare le opportune opzioni al programma di installazione; non è una operazione semplice, quindi gli utenti che installano TEX Live la installano quasi sempre completa, e fanno benissimo.

Possono incontrare problemi coloro che usano la distribuzione TEX Live per una macchina Linux conforme alle prescrizioni del consorzio Debian. A causa di queste prescrizioni e della macchinosa procedura di accettazione del software conforme alle specifiche Debian, questa distribuzione, anche se installata di default con molte varianti del sistema operativo Linux, è sempre obsoleta, talvolta anche di due anni. Nello stesso tempo quella installazione obsoleta e fortemente parziale è necessaria per soddisfare le dipendenze di molti altri programmi destinati ad un sistema Linux conforme a Debian. C'è però il modo di aggirare questo problema in modo indolore e perfettamente corretto per la sicurezza della machina. Infatti due distribuzioni TEX Live possono tranquillamente convivere nella stessa macchina senza disturbarsi a vicenda; si veda il manualetto [24] dove è specificato come fare per installare una versione di TEX Live completa accanto alla versione parziale Debian.

Chi usasse una distribuzione MiKTEX di base scoprirebbe che questa in prima installazione è predisposta per gestire la sillabazione di una mezza dozzina di lingue, fra le quali non compare l'italiano. L'utente deve allora attivare dal menù Start la pagina di dialogo dei "MiKTEX Settings"; deve cliccare sulla linguetta "Languages" dove marca con l'apposito segno le lingue di cui desidera poter usare la sillabazione (un'ottima idea è quello di marcarle tutte) e, dopo aver cliccato

sul pulsante "OK", deve selezionare la linguetta "General" ove cliccherà sul pulsante "Regenerate all formats". Questa semplice operazione (ripetibile qualora non si siano marcate tutte le lingue disponibili e se ne vogliano aggiungere altre) consente di dimenticarsi del problema dei pattern di sillabazione perché a quel punto MiKTEX è in grado di gestire tutte le lingue che conosce.

Ma se l'utente stesse scrivendo un testo sull'ostrogoto altomedievale? Ammesso che disponga dei font adatti e si sia scritto un file di descrizione di questa lingua (sulla falsa riga dei file `italian.ldf`, `latin.ldf`, eccetera), in modo da disporre di tutto il necessario per comporre in ostrogoto, scoprirebbe che il suo testo è diviso in sillabe con le regole di default, vale a dire con le regole dell'*inglese americano*.

Già! Mancano i pattern. Ma dove sono o dove trova l'utente i pattern dell'ostrogoto altomedievale? Semplicemente non esistono (almeno a tutt'oggi, 2015).

Se ha abbastanza pazienza e conosce bene la lingua ostrogota altomedievale (si suppone che lo sia, visto che sta scrivendo un testo in quella lingua) può crearsi i pattern da solo. Chi scrive l'ha fatto fin dall'inizio della sua storia con LATEX, e a suo tempo produsse la prima versione dei pattern per l'italiano, oltre che per diverse altre lingue romanze, e per il greco (classico e moderno) e per il copto (ma molto più recentemente)[2]; questo non vuol dire che egli conosca a perfezione tutte le lingue romanze e il greco classico nonché quello moderno, ma si era documentato molto bene in merito ed era riuscito nell'intento. In seguito molti utenti di TEX di madrelingua intervennero per predisporre pattern migliori, tenendo conto che nel frattempo il motore di composizione era diventato la prima approssimazione di π e che si erano resi disponibili i font latini con codifica *T1*. Per il greco gli utenti greci di TEX si erano dati un gran da fare e in pochi anni,

[2]Questo avvenne verso la fine degli anni '80, quando il programma di composizione tex gestiva ancora una sola lingua alla volta, e non esistevano i font con codifica *T1*. I francesi avevano risolto il problema creandosi una variazione della collezione dei font Computer Modern a 128 caratteri, ma che contenesse anche le lettere accentate e gli altri segni speciali che compaiono in francese, per cui avevano anche definito un sistema di pattern che facesse riferimento a quel tipo di font. In Italia non disponevamo di questa variante del font generata dagli utenti francesi, quindi gestire le lingue europee con accenti, vincolati ai font Computer Modern, era un problema non indifferente, che però si riusciva a risolvere giocando fra pattern e macro. Per l'italiano non esistevano grossi problemi perché comunque l'accento obbligatorio cade solo sull'ultima vocale delle parole tronche; per spagnolo, catalano e portoghese ci si poteva comportare come per il francese giocando fra macro e pattern; per il latino normale (non rinascimentale e non ecclesiastico) non c'erano problemi salvo il fatto che per il latino ci sarebbe voluta la sillabazione etimologica, non fonetica, e si riuscì a risolvere il problema con un compromesso fra l'etimologico e il fonetico. Per il rumeno non fu un grande successo, visto che la presenza di segni speciali è molto maggiore che nelle lingue romanze occidentali, tuttavia anche in quel caso si riuscì ad ottenere un risultato decoroso. Col greco non ci furono problemi particolari, senonché nel desiderio di avere un solo file di pattern per il greco antico politonico, per il greco moderno politonico e per il greco moderno monotonico, chi scrive aveva optato per una soluzione molto conservativa: *in dubio abstine*, e i pattern erano molto precisi, nel senso che sbagliavano sillabazione molto raramente, ma mancavano molti punti leciti di cesura.

Da allora gli unici pattern rimasti dalla prima stesura, sia pure abbondantemente riveduti e corretti, sono quelli per l'italiano e il latino; il copto arrivò quasi dieci anni dopo.

dopo la disponibilità generale dei font greci con codifica *LGR*, hanno realizzato degli ottimi file di pattern; `babel` ne carica solo due uno per il greco monotonico e un altro per le due forme di greco politonico. XeLaTeX, con il suo equivalente di `babel`, `polyglossia`, e i rinnovati file di pattern per renderli compatibili con la codifica *utf8*, caricano invece tutti e tre i file di pattern a seconda dell'opzione specificata per la lingua greca.

Solo recentemente (2014) le modifiche apportate al pacchetto `babel` e al suo modulo per la gestione del greco ha permesso di caricare effettivamente i pattern per le tre varianti del greco e di selezionarle specificando opportuni modificatori della lingua greca; a questo proposito si legga la documentazione con `texdoc babel-greek` e si leggano le documentazioni relative agli altri file per il greco citati in quel documento base. Per XeLaTeX e LuaLaTeX che usano `polyglossia` e il modulo `gloss-greek.ldf` la documentazione è contenuta tutta in quella di `polyglossia`.

Recentemente (2014) il modulo di `babel` per il latino è stato aggiornato e può fare uso di pattern distinti per il latino moderno e per quello medievale e rinascimentale basati su divisioni in sillabe su base prevalentemente fonetica, oppure per il latino classico che ha la sillabazione basata prevalentemente su regole etimologiche. Per XeLaTeX e LuaLaTeX le tre varianti sono disponibili dalla versione 2015 di TeX Live.

Questo *excursus* storico rende bene l'idea del lavoro che c'è stato dietro la produzione del necessario per comporre in circa 80 lingue diverse. Ciò nonostante può essere necessario produrre ulteriori file di pattern. Per esempio, nella seconda metà della prima decade di questo secolo fu chiesto a chi scrive di predisporre i font e i pattern per comporre nella varietà di copto, detta *bohairico*, usata nella liturgia della chiesa copta. Su richiesta di alcuni utenti, nella prima metà della seconda decade di questo secolo sono stati predisposti i pattern per comporre in retoromancio, in friulano e in piemontese.

I pattern sono molto legati alla codifica dei font di uscita; i font per il copto bohairico non trovano posto nella codifica UNICODE, ma sono traslitterabili tranquillamente con i segni della tastiera latina; i pattern furono fatti, ma per ora vengono usati solo da chi scrive e da un piccolo gruppo di persone, diaconi della chiesa copta di Gran Bretagna, che stano ricomponendo i libri liturgici, specialmente per i fedeli inglesi o naturalizzati, con il testo copto che procede parallelamente al testo inglese. Chi scrive non ha avuto difficoltà particolari nella creazione dei pattern, avendo ricevuto le relative istruzioni grammaticali da uno dei diaconi coinvolti nel progetto e con la sua verifica periodica della correttezza dei risultati.

Questi pattern non sono mai stati caricati su CTAN e quindi non sono pubblicamente disponibili. Tuttavia debbono venire ricaricati in automatico ogni volta che si aggiorna TeX Live, almeno sulle nostre macchine.

La procedura è relativamente semplice: basta caricare il file dei pattern per il bohairico `bohahyph.tex` nella cartella `.../texlive/texmf-local/tex/generic/hyphen`; contemporaneamente deve essere generato in altro file che elenca i pattern e le lingue da usare, ma che contenga solo quelle che non compaiono

già nella distribuzione standard. Detto file deve chiamarsi `language-local.dat` (o `language-local.def` per i pattern da usare con XeLaTeX codificati in *utf8*). I nostri pattern non sono codificati in modo conforme a UNICODE e sono traslitterati con lettere latine, quindi il loro nome va solamente nel file `.dat`, che conterrà quindi:

```
% Local hyphenation list language.dat
%
% Bohairic Coptic with Athanasius5 fonts
bohairic bohahyph.tex
```

A questo punto, dopo aver aggiornato il database dei nomi dei file contenuti in questo ramo locale del sistema TeX ed aver provveduto alla rigenerazione dei file di formato che devono precaricare i pattern, pdflatex funziona perfettamente anche quando si debba usare la lingua bohairica.

Ovviamente questo procedimento vale anche per i pattern del nostro ipotetico scrittore sull'ostrogoto altomedievale[3]; l'importante è che nostro ipotetico scrittore sappia costruirsi il file di pattern o sappia a chi rivolgersi per creare detto file. Si ricordi che ad ogni modifica, aggiunta o correzione del file di pattern, deve ricreare il formato almeno del programma che usa per comporre il suo testo, presumibilmente pdflatex.

Esiste un modo per creare il file di pattern senza dover conoscere la grammatica della lingua? Sì, esiste, ed è quello che ha usato Knuth o meglio, il suo assistente/dottorando Frank Liang, per generare i pattern i modo automatico. Automatico, per la verità, è una parola grossa; bisogna disporre o predisporre un elenco di parole, tante di più quante sono le eccezioni ad una regola precisa, già divise in sillabe e da dare in pasto al programma patgen, che fa già parte della distribuzione TeX Live. Purtroppo la documentazione, leggibile con `texdoc patgen` dal terminale, riguarda il programma patgen in quanto tale; come si usi è descritto in modo molto parziale e comprensibile solo a chi ci mette molta buona volontà. Tuttavia la difficoltà non risiede nel comprendere come si usi patgen, ma consiste nel generare il file sorgente di parole già divise in sillabe e traslitterate con l'alfabeto latino anche con le lettere accentate secondo la codifica *T1*. Si capisce benissimo che i pattern generati saranno tanto più efficaci quanti più casi di sillabazione sono presenti nel file di ingresso di patgen. Per i pattern validi con la codifica *utf8* la procedura è diversa, ma solitamente ci pensano i membri del gruppo di lavoro del TUG che si occupano della sillabazione.

Chi scrive ha sempre lavorato a mano per generare i file di pattern e ha usato patgen per vedere se con l'italiano si potevano ottenere meno pattern

[3]Ho inventato il nome 'ostrogoto altomedievale' tanto per fare un esempio di una lingua forse esistita, ma sicuramente improbabile; recentemente uno dei collaboratori di questa guida mi ha chiesto se me la sentivo di predisporre il necessario per comporre in *cimbro*; non ci ho ancora messo mano, anche perché non è possibile trovare una grammatica che specifichi le regole di sillabazione di questa lingua, almeno non l'abbiamo ancora trovata. Non solo, ma la mia conoscenza delle lingue portate in occidente dai goti è molto scarsa, per non dire totalmente assente; conosco pochissimo il tedesco e riconosco gli scritti in lingue alemanniche, ma una cosa è riconoscere uno scritto, un'altra completamente diversa è conoscere la lingua.

dei 330 circa che egli ha usato. In realtà il numero di pattern con patgen era marginalmente maggiore di quelli ottenuti a mano ma il numero di "trie" era maggiore, anche perché i pattern generati con patgen arrivavano fino al codice di consenso 5, mentre quelli generati a mano arrivavano solo fino a 4. Che sia una prova della superiorità della intelligenza umana rispetto all'intelligenza artificiale? Non trarrei una conclusione del genere; infatti non ho la più vaga idea di quanto debba essere grande un "dizionario" per le parole italiane già divise in sillabe e non ho idea di quanto tempo occorra per scriverlo e controllare che sia assolutamente privo di errori. Ma disponendo di un tale dizionario, lavorare con patgen può, forse, non dare luogo al miglior risultato possibile ma, disponendo del "dizionario" di parole già divise in sillabe, il lavoro per creare i pattern con patgen si completa in un paio d'ore.

Capitolo 26

Codifica in entrata e codifica in uscita

Il problema delle codifiche è delicato e spesso difficile da capire da parte di chi non conosce il funzionamento interno del proprio calcolatore. Chi vuol arrivare subito alle conclusioni operative salti direttamente all'ultimo paragrafo di questo capitolo: 26.5. Poi se vuole capire perché vengono suggerite quelle soluzioni può tornare qui all'inizio per vedere le cose tecniche che portano a quelle conclusioni.

26.1 Introduzione

L'argomento della codifica è uno di quelli che più travagliano gli utenti del sistema TEX. Per cercare di fare un po' di luce su questi argomenti, bisogna rifarsi al calcolatore che elabora i dati. Al suo interno ogni tipo di informazione è "scritto" in termini di cifre binarie '0' e '1'. Sia ben chiaro: queste cifre non sono davvero scritte nel modo che noi umani comunemente associamo al verbo 'scrivere'. I segni '0' e '1' sono essi stessi dei modi codificati per descrivere gli stati interni di certe quantità fisiche all'interno del calcolatore: saranno tensioni oppure correnti positive o negative; saranno delle piccolissime parti di materiale magnetico magnetizzato con il polo nord in alto oppure in basso; saranno delle cariche elettriche positive o negative localizzate in certi cristalli di silicio o di altro materiale semiconduttore; in realtà non ci interessa andare così a fondo nella struttura della macchina per capire che le cifre binarie '0' e '1' sono modi convenzionali per descrivere uno o l'altro dei due stati opposti che possono assumere quelle grandezze fisiche in quei determinati punti della macchina. A noi interessa il significato che di volta in volta viene attribuito a quei modi di essere di quelle grandezze fisiche.

I segnali che vengono trasmessi da un punto all'altro della macchina sono sequenze di 'zeri' e 'uni' e l'informazione che essi trasportano da un punto all'altro della macchina dipende dal trasmettitore e dal ricevitore che usano un codice per

attribuire questo significato e per comportarsi di conseguenza. Ogni 'zero' e ogni 'uno' di una sequenza si chiama *bit*, un breve acronimo diventato nome comune, ottenuto da "Binary unIT", unità binaria, e rappresenta la minima quantità di informazione che può essere contenuta in un messaggio (di un solo bit).

A livello macroscopico, quando noi battiamo un tasto sulla tastiera questa invia una sequenza di bit particolare (ovviamente diversa da tasto a tasto) all'unità centrale di elaborazione che ne farà l'uso corrispondente al ruolo che in quel momento la tastiera sta svolgendo – la tastiera non svolge solo il ruolo di strumento di scrittura. La tastiera fisica non manda all'unità centrale una lettera, ma una sequenza di bit; otto bit (oggi[1]) formano un *byte*; un byte di otto bit può assumere 256 configurazioni diverse: 00000000, 00000001, 00000010, 00000011, ..., 11111111, associabili, come numerazione binaria ai numeri decimali 0, 1, 2, 3, ..., 255. Ma, ecco, già questa è una codifica (anche se non è l'unica lettura possibile), perché a ogni sequenza di bit viene associata una lettura in termini di numero binario; insomma con codifiche diverse (modi di interpretazione diversi) lo stesso byte può assumere significati diversi.

Ma è solo una faccia della questione che riguarda la codifica. Torniamo alla nostra tastiera: la sequenza di bit che una data tastiera invia all'unità centrale è la stessa quando si preme lo stesso tasto ma, per esempio, il secondo tasto da sinistra della terza fila dall'alto significa 'w' con una tastiera QWERTY ma significa 'z' con una tastiera QZERTY. Per facilitare la lettura agli umani, sui tasti sono disegnati determinati segni, ma la tastiera fisica è del tutto indifferente a ciò che è disegnato sui tasti. Tra la tastiera e l'unità centrale di elaborazione c'è quindi un 'filtro' che traduce la sequenza di bit inviati dal tasto premuto in un segno piuttosto che nell'altro a seconda di come l'operatore ha configurato la macchina quando ha risposto ad una apposita domanda che gli ha fatto il programma di installazione del sistema operativo: "che tipo di tastiera hai?"; questo è particolarmente evidente quando si installa una distribuzione Linux, perché questo sistema operativo non è legato a nessun particolare tipo di hardware, nel senso che dovrebbe essere in grado di adattarsi a qualsiasi tipo di hardware di qualunque marca e prodotto da qualunque fabbricante.

Il filtro, comunemente chiamato *driver* di tastiera, può però essere cambiato a piacere; l'utente può installare diversi driver di tastiera sulla sua macchina, e li può rendere attivi (uno alla volta, evidentemente) in modo da poter toccare gli stessi tasti, ma inviare il segno 'w' oppure il segno σ, o qualunque altro segno di qualunque alfabeto che il driver faccia corrispondere a quel famoso secondo tasto da sinistra della terza fila dall'alto. Si tratta di codifiche diverse: cambiando driver si cambia il modo di interpretare la stessa sequenza di bit.

Nello stesso modo possiamo distinguere una codifica d'ingresso quando l'unità centrale di elaborazione (CPU, *central processing unti*) e il programma che in quel

[1] In passato il byte era definito come la più piccola sequenza di bit associabile a un indirizzo di memoria del calcolatore; questi "bocconi" di informazione potevano essere formati da 7, o da 8, o anche da 9 bit; potevano anche avere tre bit in più se erano dei byte autocorrettivi; ma questo esula completamente dall'argomento di questo testo. Basti sapere che "oggi" tutti i sistemi di calcolo usano byte di 8 bit.

momento sta "girando" ricevono una sequenza di bit da qualunque periferica capace di inviare alla CPU segnali formati da sequenze di bit, e una codifica di uscita quando sono l'unità centrale di elaborazione e il programma che vi sta girando in quel momento che inviano segnali alle periferiche. I concetti di 'ingesso' e di 'uscita' si riferiscono quindi ai segnali che entrano nella CPU per essere elaborati, oppure che vi escono dopo l'elaborazione per essere utilizzati da "terze parti".

Nel caso particolare del sistema TEX, ci ritroviamo con l'unità centrale sulla quale gira il programma che chiamiamo editor; i segnali di entrata arrivano dalla tastiera attraverso il suo driver o dal disco fisso (o da qualunque altro dispositivo di memorizzazione di massa) attraverso un altro driver specifico. L'unità centrale e l'editor che vi sta girando inviano segnali al video e al disco fisso (o altro dispositivo) ciascuno attraverso il suo specifico driver. L'editor opportunamente azionato può inviare segnali anche al sistema operativo per ordinargli di mettersi ad eseguire uno dei programmi di composizione del sistema TEX; la CPU sospende l'esecuzione dell'editor e lo mette da parte, carica e lancia l'esecuzione di uno dei programmi del sistema TEX, e assieme richiedono al disco fisso di inviare all'unità centrale i segnali corrispondenti al testo da comporre con il mark-up di LATEX; il flusso di segnali viene elaborato dalla CPU e il risultato viene inviato nuovamente al disco fisso e viene chiuso il programma di composizione richiamando dal suo riposo l'editor; la CPU invia all'editor segnali che il processo di composizione è andato a buon fine, e l'editor rilancia segnali per informare l'unità centrale che può attivare il programma di visualizzazione; il gioco ricomincia: l'unità centrale sospende l'editor e lo rimette da parte, apre o riprende il programma di visualizzazione precedentemente messo da parte e gli dice di riprendere il suo lavoro; questo programma a sua volta richiede al disco di lanciargli i segnali dei dati da visualizzare e provvede con suoi driver interni a mandare i debiti segnali al video affinché esso mostri sullo schermo il risultato della composizione; eventualmente l'operatore dà ordine all'editor di mettere in stampa il prodotto della composizione, ma sebbene le macchine periferiche siano diverse, i concetti generali di collaborazione fra le varie parti della macchina, i programmi e le periferiche è sempre lo stesso.

Non continuo perciò a descrivere questo dialogo fra sistema operativo, CPU, periferiche come la tastiera, il video e i dischi e gli altri sistemi di memorizzazione di massa, la memoria volatile interna, chiamata RAM, per tenere da parte i programmi che in quel momento sono inoperosi, perché il discorso diventerebbe lunghissimo e noiosissimo. Metto in risalto che ogni volta che si ha un dialogo fra una parte e l'altra della macchina o fra un programma e l'altro, i segnali sono in codice e ogni macchina periferica o centrale può decifrarli attraverso gli appositi driver.

Che cosa c'entra allora l'operatore/compositore in tutta questa storia di codifiche? Perché il sistema operativo non provvede da solo a gestire queste cose senza tirare in ballo quel poveraccio del compositore? Con i programmi e i sistemi chiusi questo in genere succede, per esempio con la componente Writer di LibreOffice questi problemi non si presentano; il sistema TEX non è un sistema

32	␣	48	0	64	@	80	P	96	‘	112	p
33	!	49	1	65	A	81	Q	97	a	113	q
34	"	50	2	66	B	82	R	98	b	114	r
35	#	51	3	67	C	83	S	99	c	115	s
36	$	52	4	68	D	84	T	100	d	116	t
37	%	53	5	69	E	85	U	101	e	117	u
38	&	54	6	70	F	86	V	102	f	118	v
39	’	55	7	71	G	87	W	103	g	119	w
40	(	56	8	72	H	88	X	104	h	120	x
41	)	57	9	73	I	89	Y	105	i	121	y
42	*	58	:	74	J	90	Z	106	j	122	z
43	+	59	;	75	K	91	[	107	k	123	{
44	,	60	<	76	L	92	\	108	l	124	|
45	-	61	=	77	M	93	]	109	m	125	}
46	.	62	>	78	N	94	^	110	n	126	~
47	/	63	?	79	O	95	_	111	o	127	

Tabella 26.1: I 95 caratteri ASCII stampabili. Il carattere ASCII 32 è lo spazio, qui reso visibile (␣) mediante il comando `\textvisiblespace`. Il carattere ASCII 127 (il novantaseiesimo della tabella) non è un carattere stampabile.

chiuso in sé stesso, e non solo perché è un sistema che deve poter lavorare su qualunque tipo di hardware, con qualunque periferica e sotto il controllo di qualunque sistema operativo, ma anche perché vuole poter essere totalmente staccato da simili situazioni contingenti affinché i suoi file siano trasportabili da una macchina all’altra anche con sistemi operativi eventualmente diversi.

Il sistema TEX al suo interno funziona con ulteriori codifiche sue proprie.

26.2 Le tre distinte codifiche di TEX

Nella pagina 349 si è definito ciò che si intende per codifica di un font. La definizione si riferiva ai file relativi ai font che vengono usati per ‘scrivere’ il documento nel file di uscita. Nello stesso tempo si è parlato spesso di codifica dei caratteri con cui si scrive il file `.tex` da comporre. Quindi si è parlato di codifiche di uscita e di codifiche di entrata.

Qui bisogna fare attenzione a non confondere le idee; infatti la codifica in entrata, quella che serve per creare o modificare i file `.tex`, riguarda il modo di funzionare dello *shell editor* con il quale si eseguono queste operazioni e con il quale si salva il file `.tex`. La codifica in uscita riguarda le polizze dei caratteri che si usano per la composizione. Fra queste due codifiche si trova il programma di composizione pdflatex che usa una sua codifica interna. Detto in altri termini: l’editor non si preoccupa del fatto che il testo e il mark-up che esso legge dal disco oppure dalla tastiera verrà successivamente elaborato da un programma

invece che da un altro; l'importante è che passi le informazioni giuste con il file che il successivo programma di composizione leggerà dal disco in modo che l'elaborazione dell'informazione contenuta nel disco produca la composizione tipografica desiderata. Ricordiamoci anche che lo *shell editor* con il quale creiamo o modifichiamo e salviamo il file `.tex` non fa propriamente parte del sistema TEX; questo interviene solo per trasformare il file sorgente `.tex` nel file composto `.pdf`.

Questa comunicazione fra l'editor e il programma di composizione si ottiene mediante l'invocazione dei pacchetti *inputenc* e *fontenc* con le opzioni appropriate.

Quando si invoca il pacchetto *inputenc* con una qualunque opzione di codifica che il pacchetto accetti, si stabilisce una corrispondenza fra la codifica dei caratteri immessi con la tastiera nello *shell editor* (che esso trascrive nel file `.tex` e che mostra graficamente sullo schermo), e quanto dovrà fare il successivo programma pdflatex per tradurlo nella sua codifica interna quando, leggendo il file `.tex`, ne eseguirà la composizione.

Quando invece si invoca il pacchetto *fontenc* con una qualunque opzione accettata da questo pacchetto, si stabilisce una corrispondenza fra il codice interno di pdflatex e i vari segni delle polizze dei font usati, caratterizzati dalla codifica specificata e trascritti da pdflatex sia nella pagina in lavorazione, sia nelle pagine emesse nel file di uscita, le pagine composte che finalmente saranno visualizzate sullo schermo o stampate su carta.

Si noti che i codici emessi dal driver della tastiera, o letti dal disco sotto forma di sequenze di bit, possono essere letti anche come numeri binari con i loro corrispondenti decimali, e noi possiamo identificare il carattere 'w', per esempio, con il numero decimale 119 e possiamo dire; "viene inviato il carattere 119 dalla tastiera all'editor". Il numero decimale corrisponde quindi ad una specie di indirizzo che corrisponde alla posizione della lettera 'w' in una certa tabella di caratteri corrispondente ad una certa codifica.

In sostanza la tastiera invia alla CPU delle sequenze di byte che le indicano quale tasto è stato premuto e se contemporaneamente erano stati premuti uno o più dei tasti di controllo, come il tasto delle maiuscole o quello delle maiuscole permanenti, il tasto etichettato Ctrl, o quello etichettato Alt, eccetera. In base alle impostazioni del sistema il driver della tastiera traduce questa sequenza di byte in un'altra sequenza, spessissimo costituita da un solo byte, che indica l'indirizzo del segno secondo la codifica scelta. Se si tratta di un unico byte il cui primo bit sia 0, allora il carattere corrisponde ad uno di quelli indicati nella tabella 26.1, che è comune a tutte le altre codifiche.

Se si tratta di un solo byte e il primo bit vale 1, esso può corrispondere ad un carattere diverso a seconda della codifica; nella tabella 26.2 sono riportati i vari caratteri con indirizzi compresi fra 128 e 255 corrispondenti ad alcune delle codifiche più comuni: *ansinew* adatta ai sistemi Windows (praticamente coincide con la *code page* 1252, per la quale si potrebbe specificare l'opzione *cp1252*); *latin1* adatta ai sistemi Linux e Macintosh; *latin9* adatta agli stessi sistemi ma con alcuni caratteri diversi, compreso quello dell'euro; *applemac* adatta ai

sistemi Macintosh per i quali si voglia mantenere l'uso della codifica specifica, disponibile con i sistemi operativi precedenti al Mac OS X, ma usabile anche con questo sistema operativo; *cp437* corrisponde alla codifica delle macchine che usavano il sistema operativo DOS; *cp1250* corrisponde alla codifica delle macchine Windows degli anni '90 ed è la versione usata in quei paesi dell'Europa orientale che usano varianti dell'alfabeto latino.

Nella tabella alcuni caratteri sembrano non produrre nessun risultato; uno è il carattere relativo allo spazio indivisibile (*no break space*); un altro è il marcatore di parola composta (*compound word marker*) che di fatto, con il sistema TEX, è sostituito dal punto di cesura facoltativo reso con il comando `\-`.

Come si vede non c'è praticamente nessuna differenza, tranne per la colonna intestata `latin9` per soli quattro caratteri (l'euro, № 161, la Š, № 166, la š, № 168, e il carattere № 255 che non viene usato) e quindi la codifica *latin1* o, se si preferisce, la codifica *latin9* possono essere usate con tutti e tre i sistemi operativi senza problemi, almeno per quel che riguarda i caratteri dal 161 al 255. Si consiglia la codifica *utf8* (o la codifica *latin1* – vedi nel seguito; anche perché molti dei restanti segni sono poi tutti da rendere in uscita con l'uso del pacchetto *`textcomp`* che provvede a estendere il numero di glifi disponibili con un altro centinaio di simboli alfabetici e paralfabetici).

Le altre colonne sono invece molto diverse dalla prime tre; lasciando perdere le ultime due colonne (che si riferiscono per lo più a situazioni particolari e che non riguardano la grande maggioranza degli utenti italiani che dispongono di macchine "moderne") la codifica *applemac* è abbastanza diffusa fra gli utenti delle macchine Macintosh. *Se ne scoraggia l'uso con vigore.* Scambiarsi file codificati in questo modo fra utenti che usano diverse macchine, non tutte Macintosh, vuol dire produrre fastidio e irritazione fra tutti gli autori che collaborano ad un progetto collettivo. La codifica *latin1* va benissimo per tutte le macchine moderne più diffuse; la codifica *utf8*, di cui si parlerà fra poco, va ancora meglio.

Tanto per essere espliciti, la frase:

> La città è più bella, perché così può essere più pulita.

scritta con la codifica *applemac* e aperta con un editor impostato per la codifica *latin1* appare sullo schermo così:

> La citt^ pi bella, perchŽ cos“ pu˜ essere pi pulita.

Ancora migliore è la codifica UNICODE, che è quasi universale, nel senso che si riferisce a centinaia di migliaia di segni, non solo i segni delle codifiche a 256 caratteri, e comprende ogni genere di alfabeto, ogni verso di scrittura, ogni collezione di ideogrammi e di grafemi. La lista dei segni è talmente lunga che per consultarla è meglio riferirsi alle pagine UNICODE reperibili sul sito del consorzio omonimo: `http://www.unicode.org/Public/UNIDATA/UnicodeData.txt`. I codici UNICODE possono contenere fino a 6 cifre esadecimali (3 byte), di cui la prima può assumere solo i valori 0 e 1; quindi essi vanno da `000000` a `10FFFF` (cioè 1 114 112 segni; in teoria potrebbe arrivare fino a `1FFFFF` e quindi avrebbe

spazio per 2 097 152 differenti segni). La codifica UTF-8 è un modo particolare, un algoritmo, per generare il codice UNICODE di ogni carattere in modo da usare meno byte per indirizzare i segni che occupano le prime posizioni della codifica UNICODE; a tutti gli effetti pratici possiamo identificare la codifica UTF-8 (*utf8* per il pacchetto *inputenc*) con la codifica UNICODE. Per saperne di più sulla codifica UTF-8 si può vedere il sito `http://it.wikipedia.org/wiki/UTF-8`. Con due byte solamente si possono indirizzare tutti i segni per gli alfabeti latino (anche con diacritici), greco cirillico, copto, ebraico e arabo; per tutti gli ideogrammi (cinese giapponese e coreano) occorrono tre byte; UTF-8 usa quattro byte solo per segni particolarissimi che vengono usati piuttosto raramente.

In modo analogo la codifica di uscita, quella da specificare al pacchetto *fontenc*, indica la posizione che un dato carattere occupa nel file che contiene le informazioni per disegnare il suo glifo sullo schermo, o sulla carta, o sulla periferica di uscita; la codifica di uscita, quindi, riguarda i font con cui viene composto il documento finale; non per niente quando viene composto questo capoverso l'informazione che viene memorizzata nel file di uscita, prima ancora di indicare le posizioni dei caratteri del font da usare, è `\T1/lmr/m/n/10`, cioè, in codice, l'agglutinazione delle informazioni relative (*a*) alla codifica del file che contiene le istruzioni per 'disegnare' i singoli caratteri (`T1`); (*b*) alla famiglia di caratteri (`lmr`, Latin Modern Roman); (*c*) alla serie dei caratteri (`m`, serie di media nerezza); (*d*) alla forma dei caratteri (`n`, caratteri normali diritti); (*e*) al corpo (`10`, corpo di 10 pt).

Va detto che in teoria il pacchetto per la codifica d'ingresso *utf8* potrebbe codificare più di un milione di segni; però la cosa sarebbe poco pratica e probabilmente impossibile da realizzare anche con i calcolatori più moderni che possono usare memorie di grandissime dimensioni. Per cui il pacchetto *inputenc* con l'opzione *utf8* si limita a definire solo le codifiche dei caratteri che possono venire effettivamente rappresentati con i font di uscita; per poterlo fare deve conoscere le codifiche di tutti i font che verranno usati nel corpo del documento. Per questo sarebbe opportuno caricare il pacchetto *inputenc* solo dopo aver caricato tutti gli altri pacchetti che si riferiscono ai font. In particolare sarebbe opportuno caricare prima, non solo il pacchetto *fontenc* con le opzioni che si desiderano, ma anche il pacchetto *textcomp* e ogni altro pacchetto per l'uso di collezioni di simboli speciali non compresi in quelli rappresentabili con la codifica *T1* né con la codifica *TS1* di *textcomp*.

Bisogna però ricordare che fra l'editor e il programma di composizione non c'è un collegamento diretto, ma i due programmi comunicano solo attraverso il file `.tex`.

Ecco perché questo file dovrebbe poter trasferire l'informazione della codifica di ingresso che l'editor ha usato per scriverlo al programma di composizione affinché questo possa recuperare l'informazione da trasferire correttamente nel file di uscita sotto forma di documento composto tipograficamente.

D'altro canto sarebbe opportuno che quando un file sorgente viene generato o modificato mediante un editor usando una certa codifica, esso venisse salvato su

Tabella 26.2: Alcune codifiche di ingresso per i caratteri latini

Codice	ansinew	latin1	latin9	applemac	cp437	cp1250
128	€			Ä	Ç	€
129				Å	ü	
130	‚			Ç	é	‚
131	*f*			É	â	
132	„			Ñ	ä	„
133	…			Ö	à	…
134	†			Ü	å	†
135	‡			á	ç	‡
136	ˆ			à	ê	
137	‰			â	ë	‰
138	Š			ä	è	Š
139	‹			ã	ï	‹
140	Œ			å	î	Ś
141				ç	ì	Ť
142	Ž			é	Ä	Ž
143				è	Å	Ź
144				ê	É	
145	‘			ë	æ	‘
146	’			í	Æ	’
147	“			ì	ô	“
148	”			î	ö	”
149	•			ï	ò	•
150	–			ñ	û	–
151	—			ó	ù	—
152	˜			ò	ÿ	
153	™			ô	Ö	™
154	š			ö	Ü	š
155	›			õ	¢	›
156	œ			ú	£	ś
157				ù	¥	ť
158	ž			û	Pts	
159	Ÿ			ü	*f*	ź
160				†	á	
161	¡	¡	¡	°	í	ˇ
162	¢	¢	¢	¢	ó	˘
163	£	£	£	£	ú	Ł
164	¤	¤	€	§	ñ	¤
165	¥	¥	¥	•	Ñ	Ą
166	¦	¦	Š	¶	ª	¦

Continua nella pagina seguente

Continua dalla pagina precedente

Codice	ansinew	latin1	latin9	applemac	cp437	cp1250
167	§	§	§	ß	º	§
168	¨	¨	š	®	¿	¨
169	©	©	©	©	¬	©
170	ª	ª	ª	™	½	Ş
171	«	«	«	´	¼	«
172	¬	¬	¬	¨	¡	¬
173				≠	«	
174	®	®	®	Æ	»	®
175	¯	¯	¯	Ø		Ż
176	°	°	°	∞		°
177	±	±	±	±		±
178	²	²	²	≤		˛
179	³	³	³	≥		ł
180	´	´	Ž	¥		´
181	µ	µ	µ	µ		µ
182	¶	¶	¶	∂		¶
183	·	·	·	Σ		·
184	¸	¸	ž	Π		¸
185	¹	¹	¹	π		ą
186	º	º	º	∫		ş
187	»	»	»	ª		»
188	¼	¼	Œ	º		Ľ
189	½	½	œ	Ω		˝
190	¾	¾	Ÿ	æ		ľ
191	¿	¿	¿	ø		ż
192	À	À	À	¿		Ŕ
193	Á	Á	Á	¡		Á
194	Â	Â	Â	¬		Â
195	Ã	Ã	Ã	√		Ă
196	Ä	Ä	Ä	ƒ		Ä
197	Å	Å	Å	≈		Ĺ
198	Æ	Æ	Æ	Δ		Ć
199	Ç	Ç	Ç	«		Ç
200	È	È	È	»		Č
201	É	É	É	…		É
202	Ê	Ê	Ê			Ę
203	Ë	Ë	Ë	À		Ë
204	Ì	Ì	Ì	Ã		Ě
205	Í	Í	Í	Õ		Í
206	Î	Î	Î	Œ		Î

Continua nella pagina seguente

Continua dalla pagina precedente

Codice	ansinew	latin1	latin9	applemac	cp437	cp1250
207	Ï	Ï	Ï	œ		Ď
208	Đ	Đ	Đ	–		Đ
209	Ñ	Ñ	Ñ	—		Ń
210	Ò	Ò	Ò	“		Ň
211	Ó	Ó	Ó	”		Ó
212	Ô	Ô	Ô	‘		Ô
213	Õ	Õ	Õ	’		Ő
214	Ö	Ö	Ö	÷		Ö
215	×	×	×	◊		×
216	Ø	Ø	Ø	ÿ		Ř
217	Ù	Ù	Ù	Ÿ		Ů
218	Ú	Ú	Ú	/		Ú
219	Û	Û	Û	¤		Ű
220	Ü	Ü	Ü	‹		Ü
221	Ý	Ý	Ý	›		Ý
222	Þ	Þ	Þ	fi		Ţ
223	ß	ß	ß	fl		ß
224	à	à	à	‡	α	ŕ
225	á	á	á	·	β	á
226	â	â	â	,	Γ	â
227	ã	ã	ã	„	π	ă
228	ä	ä	ä	‰	Σ	ä
229	å	å	å	Â	σ	Ĺ
230	æ	æ	æ	Ê	μ	ć
231	ç	ç	ç	Á	γ	ç
232	è	è	è	Ë	Φ	č
233	é	é	é	È	θ	é
234	ê	ê	ê	Í	Ω	ę
235	ë	ë	ë	Î	δ	ë
236	ì	ì	ì	Ï	∞	ě
237	í	í	í	Ì	ϕ	í
238	î	î	î	Ó	ε	î
239	ï	ï	ï	Ô	$\cap$	ď
240	ð	ð	ð		$\equiv$	đ
241	ñ	ñ	ñ	Ò	$\pm$	ń
242	ò	ò	ò	Ú	$\geq$	ň
243	ó	ó	ó	Û	$\leq$	ó
244	ô	ô	ô	Ù		ô
245	õ	õ	õ	ı		ő

Continua nella pagina seguente

Continua dalla pagina precedente

Codice	ansinew	latin1	latin9	applemac	cp437	cp1250
246	ö	ö	ö	ˆ	÷	ö
247	÷	÷	÷	˜	≈	÷
248	ø	ø	ø	¯	°	ř
249	ù	ù	ù	˘	∙	ů
250	ú	ú	ú	˙	•	ú
251	û	û	û	°	√	ű
252	ü	ü	ü	˚	^{n}	ü
253	ý	ý	ý	˝	^{2}	ý
254	þ	þ	þ	¸	■	ţ
255	ÿ	ÿ		ˇ		˙

disco con la stessa codifica. Sembra ovvio, ma non è così: dipende dalla qualità dell'editor.

Per esperienza chi scrive può affermare che egli conosce due editor TeXShop (specifico delle macchine Macintosh) e TeXworks (multipiattaforma) che accettano delle righe di configurazione che permettono di adattare le impostazioni del programma alle informazioni fornite con queste righe; se ne parlerà più avanti. C'è l'editor Aquamacs (solo per macchine Macintosh) che può fare più o meno le stesse cose, ma con righe di configurazione che seguono una sintassi diversa da quella dei programmi menzionati sopra; anche emacs (multipiattaforma) accetta le stesse righe di configurazione che usa Aquamacs. Esistono altresì gli editor TeXstudio e Texmaker che riescono a distinguere automaticamente se un file sorgente è stato creato o modificato con la codifica UTF-8 oppure con la codifica ISO Latin 1; in realtà sono in grado di distinguere i file codificati UTF-8 dagli altri, per cui un file che non sia UTF-8 viene classificato come ISO Latin 1. Questa capacità in generale è sufficiente, ma la codifica ISO Latin 1 non è la sola codifica dove le lettere accentate nazionali sono codificate con un solo byte ma, come si è visto nella tabella 26.2, molte codifiche hanno i caratteri nazionali codificati con un solo byte, quindi il riconoscimento ISO Latin 1 potrebbe essere sbagliato; non succede spesso perché la codifica *applemac* viene usata relativamente poco, ma questo poco è già troppo... Più avanti si vedrà come riconoscere quale sia la codifica di un file `.tex`. Con l'uscita nel 2012 della versione 2.4 di TeXstudio, la documentazione continua a dire quanto riportato sopra, ma di fatto TeXstudio è in grado di decifrare le righe magiche con la sintassi di TeXShop.

I dettagli sono molto tecnici e si omettono; non conosco nessun documento dove questi dettagli siano descritti in modo comprensibile; per quel che riguarda la codifica di uscita esiste il file `encguide.pdf`, leggibile con il solito comando `texdoc encguide`, dove vengono descritte alcune delle codifiche latine, in particolare la codifica *T1*.

Però non è particolarmente difficile leggere i file, per esempio, `latin1.def`

e `t1enc.def`[2]; il primo descrive la corrispondenza fra i caratteri della tastiera codificati secondo la norma ISO 8859-1, nota anche come ISO Latin 1, e la codifica interna di pdflatex. Il secondo descrive come si passi dalla codifica interna del sistema TEX alla codifica dei font usati, e quindi alla scelta dei glifi giusti da riportare nel file di uscita.

26.2.1 La codifica di ingresso

Se si apre il file `latin1.def`, si vede che le prime righe definiscono con `\ProvideTextCommandDefault` il significato di default di alcuni comandi interni generalmente conosciuti da parte di pdflatex; per esempio, il primo comando

```
\ProvideTextCommandDefault{\textdegree}{\ensuremath{{^\circ}}}
```

dice che se `\textdegree` non è ancora stato definito, esso va composto di default in modo matematico con un circoletto ad esponente. Naturalmente esistono dei pacchetti per i font di uscita che possono definire o ridefinire questo comando di default per produrre un segno più adeguato, disegnato apposta, ma in mancanza di questi font la definizione di default produce un risultato accettabile.

Infatti `\textdegree`, tramite il pacchetto *textcomp*, produce il segno °, mentre il circoletto ad esponente è $^\circ$, come si vede, produce un segno leggermente più grande ma è accettabilissimo.

Ci sono anche dei comandi del tipo:

```
\ProvideTextCommandDefault{\textcent}
   {\TextSymbolUnavailable\textcent}
```

che emettono un messaggio di errore se si usa il comando `\textcent`; questo succede perché il segno ¢ non è facilmente simulabile in un modo che vada bene per ogni font, codifica, famiglia, serie, forma e corpo; per produrlo ci vuole dunque l'uso di una polizza di caratteri in uscita che contenga questo segno e che perciò dia una definizione valida per `\textcent`.

Dopo altri comandi di questa specie, vengono definite le corrispondenze fra i caratteri con gli indirizzi da 161 a 255 nella codifica ISO 8859-1 e la codifica interna di pdflatex. Per esempio la definizione

```
\DeclareInputText{164}{\textcurrency}
```

specifica che introducendo con la tastiera nello *shell editor* il carattere che il driver della tastiera trasforma nel codice 164, questo produce sullo schermo il segno ¤. È vero che fra i primi comandi si dichiara che il segno corrispondente al comando `\textcurrency` non è disponibile, ma se questo comando è già stato definito o viene ridefinito in altro modo, questo segno potrebbe venire stampato nel modo giusto e non verrebbe emesso nessun avviso di simbolo non disponibile; infatti qualche riga più in alto il segno è stato stampato correttamente, perché la

[2] Si trovano entrambi nella cartella `.../tex/latex/base/`.

composizione di questo testo si affida anche al pacchetto *textcomp* che permette di usare i segni della collezione Text Companion Font illustrati nella figura 18.10.

Per i caratteri in entrata non strettamente ASCII, come le lettere nazionali si hanno poi svariate definizioni del tipo seguente, valido per la lettera 'ï' che nella codifica di entrata ISO Latin 1 occupa la posizione 239:

```
\DeclareInputText{239}{\"\i}
```

Questa dichiarazione sostituisce il carattere 'ï' contenuto nel file di entrata con quanto è contenuto nel secondo argomento, ma del quale l'utente normale non deve preoccuparsi; esso serve per porre la dieresi sulla lettera 'ı' senza puntino indicato con la macro `\i`. Oggi l'utente normale non deve preoccuparsi della 'i senza puntino' (a meno che non scriva in irlandese o in turco) perché ci pensa sempre LaTeX a usarla quando occorre, almeno in modalità testo. Si vede dunque che anche in questo caso il carattere non appartenente al sottoinsieme ASCII viene sostituito con codici interni di LaTeX.

Si noti perciò che con la codifica di entrata (le altre disponibili con il sistema TeX sono concepite nello stesso modo ma, ovviamente, con definizioni diverse, come è stato precedentemente mostrato con alcuni esempi nella tabella 26.2) ogni tasto che produca nello *shell editor* un segno diverso dai caratteri ASCII viene trasformato in un segno visibile sullo schermo e nel file `.tex` che si sta scrivendo o modificando. Quando il file `.tex` verrà letto da pdflatex ogni carattere che vi si trova viene sostituito da comandi interni di LaTeX e verrà trattato come tale, finché non verrà il momento di trasferire il testo composto nel buffer di uscita costituito dalla famosa scatola 255 riservata da TeX a questo scopo.

Si ricorda che i primi 128 caratteri di ogni codifica di ingresso corrispondono sempre ai caratteri definiti dall'*American Standard Code for Information Interchange* – ASCII, che comprendono i 32 codici per l'azionamento delle telescriventi[3], le cifre, i segni di interpunzione e l'alfabeto latino di 26 lettere non accentate, in forma maiuscola e minuscola, per un totale di $128 - 32 = 96$ segni stampabili (in realtà il centoventottesimo non è stampabile e quindi ne restano solo 95). Da una codifica ISO all'altra cambiano i secondi 128 caratteri, come si è visto con la tabella 26.2, ed è per questo che esistono diversi nomi per distinguere le varie codifiche di ingresso. Ognuno dei file `.def` relativo alla codifica di ingresso definisce i comandi testuali di default e la corrispondenza fra i codici da 128 a 255 con i comandi interni di LaTeX (in alcune codifiche gli indirizzi da 128 a 160 non ricevono nessuna definizione).

[3]Cose del passato, ma non completamente. Infatti contengono anche i codici di fine riga e di tabulazione che il sistema TeX sa trattare in modo particolare; i codici di tabulazione per l'editor serve per far spostare il cursore fino alla successiva posizione di tabulazione, ma per pdflatex quel codice viene trattato come un semplice spazio. Invece i codici di fine riga `<LF>` e `<CR>` sono usati variamente dai diversi sistemi operativi e, quindi, dagli editor che lavorano su macchine con sistemi operativi diversi. Anche questo è un tormentone, come le codifiche, ma i programmi del sistema TeX sono già predisposti per evitare all'utente di doversene preoccupare. I vari editor invece possono avere qualche problemino quando viene aperto un file composto con un codice di fine riga diverso da quello che l'editor si aspetta.

Attenzione: normalmente ogni *shell editor* salva i file con la codifica predefinita nella sua configurazione. TeXShop e TeXworks possono fare di più: possono salvare un file scritto con una certa codifica anche con una codifica diversa da quella corrispondente alla sua configurazione. Di solito questa funzionalità non è un problema, anzi può essere molto utile per cambiare codifica ad un file. Bisogna solo stare attenti a non confondersi sbagliando preferenze nel pannello di dialogo per salvare o esportare un file, oppure nella apposita finestrella in basso a destra nella cornice di TeXworks; non è comune fare questi errori, ma vale la pena segnalare la necessità di stare attenti.

26.2.2 La codifica di uscita

La codifica dei font latini *T1* nelle prime 32 posizioni contiene glifi di servizio che comprendono gli accenti ed altri segni speciali; i successivi 95 caratteri sono i caratteri ASCII; dalla posizione 127 alla posizione 255 la polizza di caratteri contiene quasi tutti i segni necessari per scrivere in tutte le lingue dell'Europa occidentale; per l'Europa orientale alcuni segni vengono creati per sovrapposizione dell'accento alla lettera base, ma sono pochissimi i segni che richiedono questo trattamento; si veda per esempio la tabella 18.5.

Il file `t1enc.def` contiene le definizioni delle corrispondenze fra i comandi interni di LaTeX e le posizioni dei singoli segni nella tabella del font di uscita. La sintassi è un po' diversa da quella usata per la codifica di ingresso, ma il significato è altrettanto comprensibile. Per esempio:

```
\DeclareTextAccent{\‘}{T1}{0}
\DeclareTextAccent{\’}{T1}{1}
```

definiscono gli accenti grave e acuto mettendo in relazione i comandi LaTeX `\‘` e `\’` con la posizione dei corrispondenti segni (0 e 1, rispettivamente) nella polizza conforme alla codifica *T1*. Di fatto questi accenti possono servire solo per essere sovrapposti a lettere che non hanno corrispondenza nella codifica del font di uscita *T1*, per esempio: ò.

Per la cediglia, che è un segno da mettere sotto, non sopra una lettera, la definizione è molto più complessa:

```
\DeclareTextCommand{\c}{T1}[1]
   {\leavevmode\setbox\z@\hbox{#1}\ifdim\ht\z@=1ex\accent11 #1%
     \else{\ooalign{\unhbox\z@\crcr
        \hidewidth\char11\hidewidth}}\fi}
```

ma si dice sempre che il comando per la cediglia `\c` trova il segno ‘cediglia’ nella posizione 11 per metterlo sotto la lettera che costituisce il suo unico argomento. Però per i segni con la cediglia presenti nella polizza codificata con la codifica

T1, come ç, ţ, ş,[4] ulteriori comandi dicono che cosa fare per trovare il segno giusto nella polizza dei font.

Successivamente vengono definite le corrispondenze fra i simboli speciali e/o accentati e gli indirizzi in cui trovare questi segni nella polizza codificata *T1*. Per esempio:

```
\DeclareTextSymbol{\ss}{T1}{255}
...
\DeclareTextComposite{\"}{T1}{i}{239}
\DeclareTextComposite{\"}{T1}{\i}{239}
```

La prima dichiarazione serve per indicare che il segno ß ottenibile con il comando interno `\ss` si trova nella posizione 255 della polizza. Invece il segno ottenuto usando il comando per la dieresi con l'argomento `i` (i con il puntino) oppure `\i` (ı senza puntino) producono entrambi la 'ï' già accentata[5] che si trova nella posizione 239 della polizza dei font codificati *T1*.

Se invece non si è specificata nessuna opzione per la codifica del font di uscita, LaTeX usa la codifica di default *OT1*, tabella 18.3; per questa codifica aprendo il file `ot1enc.def` si trovano le seguenti dichiarazioni:

```
\DeclareTextAccent{\"}{OT1}{127}
\DeclareTextSymbol{\i}{OT1}{16}
\DeclareTextCompositeCommand{\"}{OT1}{i}{\"\i}
```

La prima dichiarazione indica dove trovare nella polizza dei caratteri il segno per la dieresi; la seconda dice dove trovare nella polizza dei caratteri la 'i' senza puntino; la terza dice che se in entrata si è scritto `\"i` invece che direttamente `ï`, la sequenza per il comando della dieresi seguita dalla lettera i col puntino deve essere interpretata alla stessa maniera di come si interpreta il codice di ingresso `ï`, cioè come `\"\i`. Come si vede, entrambe le dichiarazioni sono valide solo per la codifica *OT1* dei font di uscita.

Quando quindi LaTeX incontra una 'i' accentata o con la macro per l'accento o rappresentata direttamente con il codice d'entrata per la 'i' accentata, si ritrova a dover eseguire l'espressione `\"\i`; questa si traduce nei comandi primitivi che formano la stringa `\accent127\char16`; il comando primitivo `\accent` serve per sovrapporre il segno che ha l'indirizzo specificato come suo argomento sul segno seguente che qui è indicato con il comando primitivo `\char` seguito dall'indirizzo specifico della 'i' senza puntino. Quei comandi primitivi verranno eseguiti solo quando il capoverso sarà stato elaborato sotto forma di righe giustificate con le eventuali cesure in fin di riga, pronto per essere conservato nella pagina in costruzione; ma quando l'algoritmo di divisione in righe entra in gioco, agisce sul

[4] Questi ultimi due, se usati in rumeno, non dovrebbero avere la cediglia, ma una virgola; spesso però i rumeni devono accontentarsi di una approssimazione, perché molti font mancano del segno con la virgola; esiste il pacchetto *combelow* per risolvere con LaTeX questo problema.

[5] Questo fatto succede con quasi tutti gli accenti quando l'argomento è una 'i'. Per cui diventa obsoleta la prescrizione che per mettere un accento sopra la 'i' bisogna usare *solo* la variante senza puntino.

testo che contiene ancora questi comandi primitivi. Come tali non fanno parte della stringa di token leciti in una parola, per cui l'algoritmo di sillabazione termina la 'parola' subito prima del comando per l'accento, cosicché non riesce a sillabare completamente l'intera stringa che gli umani considerano una parola.

Ora dovrebbe quindi essere chiaro che:

1. Quando pdflatex deve eseguire la sillabazione usando per l'uscita la codifica *T1* si trova nel suo buffer interno il capoverso da dividere in righe composto solo da caratteri, e non contiene comandi di nessun genere (tranne quelli che non riguardano il testo); e quindi funziona a dovere. Invece con la codifica *OT1*, quella di default, ogni comando per inserire gli accenti resta nel buffer del capoverso e abbiamo visto che questo impedisce a pdflatex di riconoscere come parole le stesse stringhe di caratteri che gli umani considerano come tali.

2. I segni accentati delle polizze codificate *T1* sono disegnati individualmente quindi lo stesso accento è collocato sulla lettera di base nella posizione più opportuna in relazione alla sua forma. Invece con la codifica *OT1* lo stesso accento viene sempre centrato sulla mezzeria della lettera di base senza tenere conto delle asimmetrie ottiche associate a ciascuna lettera di base.

Queste sono due ottime ragioni per non usare mai i font con codifiche diverse da *T1* quando si compongono con pdflatex testi in lingue che si scrivono in caratteri latini e che contengono diversi diacritici; in realtà questa raccomandazione vale anche per l'inglese, che solitamente non usa diacritici, ma li usa (*a*) nelle citazioni di nomi o parole straniere, e (*b*) nelle edizioni storiche o critiche di testi scritti in *middle English*.

Discorsi simili si possono fare per la codifica *utf8*, solo che ogni carattere UNICODE che si introduce con la tastiera attraverso lo *shell editor* nel file `.tex` non è formato da un solo byte, ma da un codice formato spesso da diversi byte, per cui la gestione dell'immissione dei caratteri nel file `.tex` è un poco più complicata; per fortuna ci pensa l'editor e l'utente è sollevato dalla necessità di occuparsi di questi dettagli. Anche la trasformazione dei caratteri immessi con eventuali comandi TEX o LATEX diventa un poco più complessa, ma ci pensa il programma di composizione. L'uso della codifica *utf8* offre però il vantaggio che, tastiera permettendo, qualunque segno immissibile con la tastiera può essere fatto corrispondere con il segno giusto, compatibilmente con la codifica dei font di uscita, cioè con l'effettiva presenza di quel segno nella polizza dei font di uscita. Se anche questi font sono codificati UNICODE, non c'è bisogno di nessun passaggio intermedio, ed è questo quello che fanno i compilatori xetex, luatex e context-mk-iv.

26.3 Specificare la codifica giusta

Qualsiasi sia la codifica dei caratteri di ingresso, nasce un problema che ogni utente di LATEX, o degli altri programmi del sistema TEX, deve affrontare.

Egli deve specificare (mediante l'uso dell'opzione giusta al pacchetto *inputenc* nel preambolo del suo documento) quale sia la codifica con il quale ha scritto o modificato il file sorgente, altrimenti il suo file non potrà essere correttamente compilato.

Un errore comune è quello di scordarsi addirittura di invocare il pacchetto *inputenc*, non parliamo di specificare una opzione valida; oppure di invocare il pacchetto ma di specificare una opzione non coerente con le impostazioni dello *shell editor*. Succede più spesso di quanto non si creda. Tizio scrive un file sorgente `.tex` usando la codifica di ingresso *utf8*; poi spedisce il file al suo collega Caio. Caio, il cui *shell editor* è configurato per usare la codifica *latin1*, apre il file inviatogli da Tizio, ma trova il testo intercalato con caratteri strani. Questa è un situazione frequentissima.

È successo anche a chi scrive (utente di calcolatori Macintosh) di ricevere da un'altra persona, anch'essa utente di un Macintosh, un file LaTeX codificato con la codifica MacRoman (*applemac*); l'altra persona non si rendeva conto che quella è una codifica ristretta ai sistemi Mac e non è diffusa come le codifiche 'apolidi', non legate ad un particolare sistema operativo o a una marca di hardware. Chi scrive abitualmente usa la codifica ISO Latin 1 per scrivere in italiano e la codifica UTF-8 per scrivere con altri alfabeti; quando ha ricevuto il file ci ha messo un poco a rendersi conto di questa situazione e quando se ne è accorto era troppo tardi per porvi rimedio in modo corretto. Quanto segue, oltre a descrivere la situazione relativa ai malintesi generati dalle varie codifiche, serve anche per indicare alcuni modi corretti per affrontare questo tipo di problemi.

Se Tizio e Caio stanno usando lo *shell editor* multipiattaforma TeXworks e se hanno configurato bene il file `.tex` non devono fare nulla. Per configurare bene il file bisogna che esso, prima del testo vero e proprio, contenga delle righe di configurazione costituite da righe di commento, scritte in modo speciale, che dicono a TeXworks se esiste e come si chiama il master file e quale sia la codifica usata per scrivere il testo (nell'esempio che segue la prima riga è facoltativa, nel senso che se il file è il master file, non c'è nessun bisogno di fornire questa informazione lapalissiana)[6]:

```
%␣!TeX␣root␣=␣MyMasterFile.tex
%␣!TeX␣encoding␣=␣IsoLatin
%␣!Tex␣TS-program␣=␣pdflatex
```

In questo modo, se Tizio e Caio usano *shell editor* diversi, ma che siano in grado di decifrare queste speciali righe di commento, sono gli editor che si auto-configurano di conseguenza e i due tastieristi non devono fare assolutamente nulla.

Esistono poi alcuni *shell editor* che riconoscono automaticamente la codifica usata per creare i file. Apparentemente Aquamacs è uno di questi *shell editor*; ho provato a far leggere ad Aquamacs un file salvato con la codifica *applemac*

[6]Le righe indicate qui di seguito seguono la sintassi di TeXShop; TeXworks ha un'altra sintassi, ma capisce anche la sintassi di TeXShop.

(MacRoman) e sullo schermo è apparso il file con le lettere accentate sostituiti dai loro codici ottali preceduti dal backslash; quindi a stretto rigore Aquamacs riconosce automaticamente molte codifiche, ma non riconosce "Western MacRoman" sebbene Aquamacs sia un programma specifico per le macchine Macintosh[7]; tuttavia la sostituzione globale di quei codici numerici ottali con le corrispondenti lettere accentate è molto facile.

L'editor Aquamcas dispone di altre righe speciali che seguono un'altra sintassi:

```
% -*- mode: latex; coding: latin-1;  TeX-master: MyMasterFile.tex -*-
```

Questa riga speciale deve essere all'inizio del file `.tex`; i due insiemi di righe speciali, quelle per Aquamacs (valide anche per emacs) e quelle per TEXShop/ TeXworks, possono coesistere nello stesso file in questo modo:

```
% -*- mode: latex; coding: latin-1;  TeX-master: MyMasterFile.tex -*-
% !TeX root = MyMasterFile.tex
% !TeX encoding = IsoLatin
% !Tex TS-program = pdflatex
```

e gli editor che le possono interpretare lo fanno senza confondersi con le righe presenti per editor diversi.

L'editor TeXstudio nelle versioni precedenti alla 2.4 è in grado di distinguere la codifica *utf8*, e classifica come ISO Latin 1 ogni altra codifica; il che può essere errato e bisogna intervenire a mano. A partire dalla versione 2.4 TeXstudio è in grado di interpretare le righe magiche e anche di convertire la codifica a scelta dell'utente.

Per impostare le righe speciali con TeXworks bisogna procedere a mano. Con TEXShop è più semplice perché il menù `Macros` contiene esplicitamente le voci `Encoding`, `Program` e `Root`; scegliendo ciascuna di queste voci si aprono pannelli di dialogo fra cui scegliere l'informazione giusta. Per Aquamacs alcune "variabili" della riga speciale devono essere introdotte a mano, ma per il nome del master file si può scegliere dal menù `LaTeX/Multifile/Parsing/Set Master File` che apre un pannello di dialogo dove scegliere il nome del file che deve svolgere il ruolo di master file.

26.3.1 Scoprire la codifica di input

Se Tizio e Caio non si trovano in queste condizioni, che cosa deve fare Caio per poter elaborare il testo contenuto nel file ricevuto da Tizio? Ha due possibilità: (*a*) cambia codifica al file `.tex`, oppure (*b*) cambia l'impostazione della codifica del suo *shell editor*. Ma prima di tutto deve scoprire quale sia la codifica del file ricevuto da Tizio.

La soluzione (*a*) sarebbe preferibile; e sarebbe preferibile anche che entrambi usassero sempre la codifica *utf8*. Se Tizio poteva compilare correttamente il suo

[7] In effetti dando ad Aquamacs il comando per esporre le codifiche che può riconoscere, la codifica Western MacRoman non è elencata.

file, era perché aveva specificato l'opzione giusta al pacchetto *inputenc*; Caio, allora, benché non riesca a leggere il testo del documento sorgente con facilità, può però leggere perfettamente il preambolo; basta allora che cerchi la stringa `inputenc` e lì trova che Tizio ha usato la codifica *latin1*. Chiude il file appena aperto, *ma senza salvarlo*, telefona a Tizio ricordandogli che avevano convenuto di usare entrambi la codifica *utf8*, riapre il file con TeXworks e procede come spiegato nel paragrafo 26.5 salvando il file con la codifica *utf8*.

Fin qui va bene. Ma se Tizio ha usato la codifica *latin1* e si è scordato di specificare la codifica del suo *shell editor* nel preambolo del documento? Be', vuol dire due cose: primo, non poteva compilare il suo file producendo il file di uscita in modo corretto e, secondo, Caio deve tirare a indovinare per scoprire la codifica usata da Tizio, probabilmente in modo inconsapevole; è chiaro che basterebbe una telefonata o un messaggio di posta elettronica; è altrettanto chiaro che il procedimento indicato nel paragrafo 26.5 permette di scoprire sia la codifica usata sia di eseguire la conversione. Ma talvolta il file è stato scritto da un anonimo Tizio che l'ha messo in rete e Caio l'ha trovato interessante guardandolo sullo schermo; ma dopo averlo scaricato non riesce più a leggerlo. In questo caso disperato, sempre senza modificare e salvare il file appena scaricato, tira ad indovinare; l'indovinello è facile se il file era stato scritto usando la codifica *utf8*, perché ogni carattere non ASCII è sostituito da due o più caratteri strani. Invece l'indovinello può essere difficile se appaiono caratteri strani isolati, specialmente alla fine delle parole. Non resta che seguire la procedura indicata nel paragrafo 26.5 e non ci sono più problemi.

Il caso più disperato si ha quando Tizio e/o Caio si servono di *shell editor* vecchiotti, che usano una sola codifica, ma purtroppo si tratta di codifiche diverse. Non c'è quindi modo di modificare le impostazioni dello *shell editor* e non rimane altra scelta che quella di cambiare, mediante appositi programmi da linea di comando, la codifica del file `.tex`. O meglio, se i due collaboratori sanno di essere in queste condizioni avranno cura di scrivere o modificare i file `.tex` sorgente del loro lavoro cooperativo usando solamente caratteri ASCII; in sostanza non useranno direttamente i caratteri nazionali della loro tastiera, ma scriveranno le varie sequenza per gli accenti come richiesto dalla sintassi originale di TEX: per l'italiano useranno perciò `\‘a`, `\‘e`, `\’e`, `\‘i`, `\‘o`, `\‘u`, e si comporteranno in modo simile con le lettere maiuscole. È chiaro che questo modo di procedere, formalmente corretto, non è molto efficiente e i due collaboratori potrebbero accordarsi per aggiornare il loro sistemi TEX e per usare lo stesso fra uno dei vari *shell editor* multipiattaforma. In ogni caso seguendo la procedura indicata nel paragrafo 26.5 possono entrambi convertirsi reciprocamente i file nel modo che vogliono. In fondo TeXworks è multipiattaforma e non è legato a nessuna particolare distribuzione del sistema TEX. Quindi se TeXworks non fosse già installato, basta che entrambi i collaboratori lo installino. Possono anche usarlo per lavorare sui loro file, non solo per scoprire la codifica oppure per convertirne la codifica; è un editor semplice, ma ha diverse caratteristiche molto interessanti.

Si tenga presente che il modo apparentemente primitivo di scrivere i caratteri accentati secondo la sintassi TEX, consente molte più combinazioni di quelle che

consentirebbe la codifica UNICODE, perché ogni segno diacritico di cui TEX dispone (una dozzina) può essere messo su qualunque lettera latina (26) minuscola o maiuscola, con un totale di combinazioni enorme, che consentono anche delle combinazioni "impossibili", cioè segni che non vengono usati in nessuna lingua; questo meccanismo non è affatto tanto primitivo come si potrebbe pensare, tant'è che costituisce la codifica interna del sistema TEX, indipendentemente dalla codifica di ingresso e da quella relativa ai font di uscita.

26.3.2 Cambiamento della codifica

Ma se proprio si deve cambiare codifica e, per qualche motivo, non si volesse installare TeXworks, tra i programmi disponibili, quello base è è l'applicativo iconv per macchine Linux e Mac OS X, la cui sintassi è:

```
iconv [-c] [-s] [-f codifica] [-t codifica] inputfile
```

dove le opzioni `-c` e `-s` controllano i messaggi d'errore nel caso che il file di ingresso contenga caratteri non convertibili. Il file di ingresso `inputfile` è scritto con la codifica specificata con l'opzione `-f`, mentre il file di uscita è scritto con la codifica specificata con l'opzione `-t`. Il programma di default scrive il file di uscita nello stream di scrittura standard (generalmente lo schermo); se si vuole scrivere su un file vero e proprio bisogna reindirizzare l'uscita verso un file tramite il codice di reindirizzamento:

```
iconv [-c] [-s] [-f codifica] [-t codifica] inputfile > outputfile
```

dove il nome del file di uscita, `outputfile`, deve essere diverso dal nome del file di entrata.

Per conoscere le codifiche disponibili sulla particolare macchina sulla quale si sta lavorando e per scrivere i nomi delle codifiche con l'ortografia giusta, conviene dare il comando:

```
iconv -l
```

e sul terminale appare tutto l'elenco delle codifiche disponibili per le quali è possibile eseguire le conversioni. Le codifiche scritte sulla stessa riga, stando alla pagina del manuale, sono sinonimi l'una dell'altra. Nella macchina sulla quale sto lavorando le codifiche disponibili (con le righe ripiegate e rientrate per i limiti della giustezza del testo) sono:

```
ANSI_X3.4-1968 ANSI_X3.4-1986 \textsc{ascii} CP367 IBM367 ISO-IR-6
    ISO646-US ISO_646.IRV:1991 US US-\textsc{ascii} CS\textsc{ascii}
UTF-8
ISO-10646-UCS-2 UCS-2 CSUNICODE
UCS-2BE UNICODE-1-1 UNICODEBIG CSUNICODE11
UCS-2LE UNICODELITTLE
ISO-10646-UCS-4 UCS-4 CSUCS4
```

```
UCS-4BE
UCS-4LE
UTF-16
UTF-16BE
UTF-16LE
UTF-32
UTF-32BE
UTF-32LE
UNICODE-1-1-UTF-7 UTF-7 CSUNICODE11UTF7
UCS-2-INTERNAL
UCS-2-SWAPPED
UCS-4-INTERNAL
UCS-4-SWAPPED
C99
JAVA
CP819 IBM819 ISO-8859-1 ISO-IR-100 ISO8859-1 ISO_8859-1
    ISO_8859-1:1987 L1 LATIN1 CSISOLATIN1
ISO-8859-2 ISO-IR-101 ISO8859-2 ISO_8859-2 ISO_8859-2:1987
    L2 LATIN2 CSISOLATIN2
ISO-8859-3 ISO-IR-109 ISO8859-3 ISO_8859-3 ISO_8859-3:1988
    L3 LATIN3 CSISOLATIN3
ISO-8859-4 ISO-IR-110 ISO8859-4 ISO_8859-4 ISO_8859-4:1988
    L4 LATIN4 CSISOLATIN4
CYRILLIC ISO-8859-5 ISO-IR-144 ISO8859-5 ISO_8859-5
    ISO_8859-5:1988 CSISOLATINCYRILLIC
ARABIC ASMO-708 ECMA-114 ISO-8859-6 ISO-IR-127 ISO8859-6
    ISO_8859-6 ISO_8859-6:1987 CSISOLATINARABIC
ECMA-118 ELOT_928 GREEK GREEK8 ISO-8859-7 ISO-IR-126 ISO8859-7
    ISO_8859-7 ISO_8859-7:1987 CSISOLATINGREEK
HEBREW ISO-8859-8 ISO-IR-138 ISO8859-8 ISO_8859-8 ISO_8859-8:1988
    CSISOLATINHEBREW
ISO-8859-9 ISO-IR-148 ISO8859-9 ISO_8859-9 ISO_8859-9:1989
    L5 LATIN5 CSISOLATIN5
ISO-8859-10 ISO-IR-157 ISO8859-10 ISO_8859-10 ISO_8859-10:1992
    L6 LATIN6 CSISOLATIN6
ISO-8859-13 ISO-IR-179 ISO8859-13 ISO_8859-13 L7 LATIN7
ISO-8859-14 ISO-CELTIC ISO-IR-199 ISO8859-14 ISO_8859-14
    ISO_8859-14:1998 L8 LATIN8
ISO-8859-15 ISO-IR-203 ISO8859-15 ISO_8859-15 ISO_8859-15:1998
ISO-8859-16 ISO-IR-226 ISO8859-16 ISO_8859-16 ISO_8859-16:2000
KOI8-R CSKOI8R
KOI8-U
KOI8-RU
CP1250 MS-EE WINDOWS-1250
CP1251 MS-CYRL WINDOWS-1251
```

```
CP1252 MS-ANSI WINDOWS-1252
CP1253 MS-GREEK WINDOWS-1253
CP1254 MS-TURK WINDOWS-1254
CP1255 MS-HEBR WINDOWS-1255
CP1256 MS-ARAB WINDOWS-1256
CP1257 WINBALTRIM WINDOWS-1257
CP1258 WINDOWS-1258
850 CP850 IBM850 CSPC850MULTILINGUAL
862 CP862 IBM862 CSPC862LATINHEBREW
866 CP866 IBM866 CSIBM866
MAC MACINTOSH MACROMAN CSMACINTOSH
MACCENTRALEUROPE
MACICELAND
MACCROATIAN
MACROMANIA
MACCYRILLIC
MACUKRAINE
MACGREEK
MACTURKISH
MACHEBREW
MACARABIC
MACTHAI
HP-ROMAN8 R8 ROMAN8 CSHPROMAN8
NEXTSTEP
ARMSCII-8
GEORGIAN-ACADEMY
GEORGIAN-PS
KOI8-T
MULELAO-1
CP1133 IBM-CP1133
ISO-IR-166 TIS-620 TIS620 TIS620-0 TIS620.2529-1 TIS620.2533-0
    TIS620.2533-1
CP874 WINDOWS-874
VISCII VISCII1.1-1 CSVISCII
TCVN TCVN-5712 TCVN5712-1 TCVN5712-1:1993
ISO-IR-14 ISO646-JP JIS_C6220-1969-RO JP CSISO14JISC6220RO
JISX0201-1976 JIS_X0201 X0201 CSHALFWIDTHKATAKANA
ISO-IR-87 JIS0208 JIS_C6226-1983 JIS_X0208 JIS_X0208-1983
    JIS_X0208-1990 X0208 CSISO87JISX0208
ISO-IR-159 JIS_X0212 JIS_X0212-1990 JIS_X0212.1990-0 X0212
    CSISO159JISX02121990
CN GB_1988-80 ISO-IR-57 ISO646-CN CSISO57GB1988
CHINESE GB_2312-80 ISO-IR-58 CSISO58GB231280
CN-GB-ISOIR165 ISO-IR-165
ISO-IR-149 KOREAN KSC_5601 KS_C_5601-1987 KS_C_5601-1989
```

```
    CSKSC56011987
EUC-JP EUCJP EXTENDED_UNIX_CODE_PACKED_FORMAT_FOR_JAPANESE
    CSEUCPKDFMTJAPANESE
MS_KANJI SHIFT-JIS SHIFT_JIS SJIS CSSHIFTJIS
CP932
ISO-2022-JP CSISO2022JP
ISO-2022-JP-1
ISO-2022-JP-2 CSISO2022JP2
CN-GB EUC-CN EUCCN GB2312 CSGB2312
CP936 GBK
GB18030
ISO-2022-CN CSISO2022CN
ISO-2022-CN-EXT
HZ HZ-GB-2312
EUC-TW EUCTW CSEUCTW
BIG-5 BIG-FIVE BIG5 BIGFIVE CN-BIG5 CSBIG5
CP950
BIG5-HKSCS BIG5HKSCS
EUC-KR EUCKR CSEUCKR
CP949 UHC
CP1361 JOHAB
ISO-2022-KR CSISO2022KR
```

Perciò per trasformare il file `myfile.tex` scritto con la codifica *latin1* in un altro file `myfile-utf8.tex` scritto con la codifica *utf8* bisogna dare il comando:

```
iconv -f LATIN1 -t UTF-8 myfile.tex > myfile-utf8.tex
```

Per convertire un file `myfilew.tex` scritto su una macchina Windows con la *code page* di default `1252` in un file con la codifica *utf8* si scriverà:

```
iconv -f WINDOWS-1252 -t UTF-8 myfilew.tex > myfile-utf8.tex
```

Per il Mac esiste un interfaccia grafica charco che permette di fare sostanzialmente le stesse cose lavorando prevalentemente con il mouse.

Per le macchine Windows si può usare ancora iconv ma bisogna installarlo dentro l'ambiente di simulazione CygWin operando come se si stesse installando il programma su una macchina Linux.

Alternativamente si può scaricare la libreria `libiconv` dal sito `http://sourceforge.net/projects/gnuwin32/files/libiconv/` e il programma di installazione provvede anche a rendere disponibile la versione Windows Win-iconv con la quale si può operare senza ricorrere all'ambiente di simulazione CygWin.

Se tutto ciò dovesse 'disturbare' o comunque apparire troppo complicato, si riesamini la possibilità di sfruttare le capacità dello *shell editor* TeXworks, oppure si ricorra alla sintassi TEX che non fa uso di caratteri speciali ma solo caratteri ASCII e usa macro particolari per indicare quali segni diacritici usare.

26.4 Collage di contributi diversi

Un altro problema sorge invece quando si compone un testo eseguendo un collage di contributi di varie persone che hanno fornito file in formati come `.doc` o `.pdf`, e il compositore procede a mano copiando dal file esterno e incollando nel file `.tex`.

In questi casi spesso sono i caratteri di interpunzione quelli che differiscono sensibilmente dai caratteri della codifica che si vorrebbe usare, fosse essa anche la codifica più generale *utf8*. A occhio nudo spesso questi caratteri (apostrofo, virgolette semplici, virgolette diritte, eccetera) sono difficili da rilevare perché il riconoscimento corretto dipende anche dal disegno dei font che vengono usati dallo *shell editor* nella finestra del file sorgente; io preferisco i font a spaziatura fissa in questa finestra del file sorgente ma, nonostante questo, quei segni sono difficili da rilevare ad occhio nudo. Tenendo conto che un file di ingresso deve essere scritto tutto con la stessa codifica, bisogna per forza convertire a mano tutti i segni nella codifica principale, oppure marcare il file sorgente con la codifica specifica usata dal collaboratore esterno o che il suo programma di trasformazione nel file `.pdf` ha usato.

TeXworks e TEXShop| potrebbero interpretare correttamente questa informazione e Aquamacs potrebbe diagnosticare da solo la codifica usata. In ogni caso si può eseguire la compilazione con pdflatex e poi andare a controllare un certo numero di istanze di questi caratteri speciali passando dalla finestra del file sorgente alla finestra del file composto sfruttando la capacità dello *shell editor* e del visualizzatore del file composto di eseguire correttamente la ricerca diretta e inversa. Se questi caratteri un po' particolari sono tutti presenti in modo corretto nel file di uscita, allora la codifica dichiarata o determinata è corretta. Altrimenti è opportuno procedere ad una correzione manuale di tutti questi segni; solitamente la ricerca e sostituzione di *tutte* le istanze di un dato carattere può essere fatta con pochi click di mouse con qualunque *shell editor*.

26.5 Considerazioni riassuntive

- La codifica di ingresso deve essere sempre specificata come opzione nell'invocazione del pacchetto *`inputenc`*; questa codifica riguarda essenzialmente l'interazione fra la tastiera e lo *shell editor* e come questo salva il file sorgente sul disco.

- Per la perfetta composizione del documento finale bisogna sempre usare font il più possibile completi, comunque almeno con la codifica *T1* per migliorare la divisione in sillabe in fin di riga e per avere i segni accentati di fattura migliore rispetto a quelli che si potrebbero avere sovrapponendo l'accento alla lettera di base.

- È fortemente consigliabile scrivere le righe di autoconfigurazione per lo *shell editor*; questo vale anche se non si usa uno *shell editor* che le capisca e possa

autoconfigurasi di conseguenza; esse rappresentano un buon promemoria permanente e sono utilissime per le altre persone a cui eventualmente si debba inviare il file sorgente.

- Se si deve cambiare codifica, allora conviene procedere come segue.
  - È necessario eseguire subito una copia del file di cui si deve cambiare la codifica; nel caso ci si sbagli con le operazioni successive, il file originale può sempre essere recuperato.
  - Se aprendo un file `.tex` con lo *shell editor* si vede il file che contiene dei caratteri strani, allora l'impostazione della codifica dello *shell editor* non corrisponde alla codifica del font con cui è stato scritto il file `.tex`. In questo caso si eviti di fare qualunque modifica al file e ci si guardi bene dal "salvarlo" su disco.
  - Si riapra il file con TeXworks; questo editor è disponibile per tutte le piattaforme e solo con le installazioni Linux potrebbe essere necessario installarlo espressamente. Per le operazioni che seguono è essenziale usare TeXworks, poi volendo si può tornare a qualunque altro editor che l'utente preferisca rispetto a TeXworks.
  - Una volta aperto il file che contiene caratteri strani, ci si porti su una pagina in cui se ne vedono alcuni, sostituiti, generalmente, da rombi neri contenenti all'interno un punto interrogativo bianco; si mantenga quella pagina.
  - Nella riga inferiore della schermata di TeXworks ci sono in basso a destra delle finestrine, nella seconda delle quali c'è la codifica impostata di default per questo editor; cliccando su questa finestra si apre il lungo elenco di codifiche che TeXworks è capace di gestire; si scelga un'altra codifica: per esempio, se inizialmente nella finestrina c'era scritto UTF8, si scelga ISO8859-1, poi si clicchi di nuovo sulla finestrina e nella prima riga in alto si scelga di "ricaricare lo stesso file con la codifica appena scelta".
  - Il file viene ricaricato: se i rombi neri spariscono, quella appena scelta è la codifica con cui era stato scritto il file; se i rombi neri non spariscono ripetere scegliendo un altra codifica come indicato nel passo precedente. Si noti che le codifiche più frequenti sono ISO8859-1, MacRoman, Windows 1250, oltre a UTF-8 che è la codifica preimpostata di TeXworks.
  - Se si vuole cambiare codifica (come si è supposto all'inizio di questa elencazione), si scelga la codifica desiderata nella solita finestrina e si salvi il file.
- In questo modo si è scoperta la codifica del file iniziale e lo si è convertito nella codifica desiderata; se si dispone di TeXShop si apra il file appena salvato e con i suoi pulsanti gli si faccia scrivere la riga magica dell'encoding

corrispondente alla codifica appena usata per la conversione in testa al file. In questo modo non ci si dovrà più preoccupare di capire/indovinare la codifica ignota di quel file.

Quando ho cominciato a curare questo testo le opzioni per usare la codifica *utf8* erano appena agli inizi e non erano sufficientemente affidabili; perciò avevo sempre consigliato di usare la codifica *latin1*; pur non essendo completamente conforme con qualunque impostazione di sistema operativo e/o di *shell editor*, per scrivere in italiano e nella maggior parte delle lingue occidentali questa codifica va benissimo; oggi che la codifica *utf8* è completamente non ho dubbi a raccomandare, più che consigliare, di usare questa codifica "universale".

Con la codifica di entrata *utf8* il file sorgente aumenta leggermente di dimensioni ma si è sicuri che, pur di usare uno *shell editor* recente che accetti la codifica di ingresso *utf8*, non esistono problemi di codifica a livello di pdflatex. Va anche notato che le versioni aggiornate dei più moderni *shell editor* già dalla prima installazione sono preimpostati per lavorare con la codifica *utf8*.

A chi si avvicina al sistema TEX per la prima volta, oppure aggiorna la propria distribuzione del sistema TEX dopo anni di mancati aggiornamenti, consiglio di installare, se non lo sono già, uno o più degli editor TeXworks, TeXstudio, TeXmaker, che sono multipiattaforma, e di impostarli, se non lo fossero già, per lavorare con la codifica di ingresso *utf8* e di mantenere sempre questa codifica. Se capitasse loro di dover aprire un file ricevuto da terzi o scaricato dalla rete che non fosse scritto con questa codifica, procedano come detto sopra per convertire il file alla codifica *utf8*.

Tuttavia la codifica *utf8* non garantisce di evitare problemi quando si esegue lavoro cooperativo, visto che i vari 'tastieristi' non necessariamente dispongono degli stessi *shell editor* operanti su macchine dello stesso genere, controllate dallo stesso sistema operativo. Bisogna sempre essere pronti a riconoscere i problemi della codifica di entrata e saperli risolvere. Questa raccomandazione è particolarmente indicata per coloro che si sono avvicinati al sistema TEX da poco; oppure a coloro che aggiornano il loro strumenti software a versioni più moderne e non sono pronti ad affrontare questo genere di novità.

Capitolo 27

Come fa TEX a comporre le pagine

Sul Forum del GUIT compaiono spesso domande che rivelano la difficoltà degli utenti a padroneggiare il meccanismo di composizione asincrona del sistema TEX. Questo meccanismo, si badi bene, è sostanzialmente identico con qualunque dei motori ci composizione del sistema, sia esso tex, pdftex, context, luatex, xetex, o gli altri motori meno conosciuti o meno usati. Per semplicità nel seguito ci riferiremo a pdftex, che permette di sfruttare la microgiustificazione, ma quanto verrà detto per questo motore di composizione vale sostanzialmente anche per gli altri, eccetto per quel che riguarda la microgiustificazione.

In tutti i casi pdftex determina per ogni capoverso e per ogni pagina dei fattori di merito o di demerito tenendo conto di misure di 'bruttezza', nel gergo di TEX chiamata *badness*, delle penalità (*penalty*) e dei fattori di demerito (*demerit*) che vengono intercalati nel testo e nelle pagine da comporre, diciamo così, all'insaputa del compositore.

Il compositore inserisce fattori di merito o di demerito o penalità di vario genere nel file sorgente quanto usa comandi quali `\pagebreak`, `\clearpage` o `\finalhyphendemerits`, ma spesso questi comandi e i relativi parametri sono già presenti nei file di classe. In più pdftex determina altri fattori di bruttezza o di demerito per conto proprio. Il risultato di tutte queste operazioni è che il motore di composizione sceglie le divisioni in righe dei singoli capoversi e le divisioni in pagine dei singoli capitoli in modo da minimizzare la bruttezza complessiva dei capoversi oppure delle pagine.

L'algoritmo che determina questo minimo è decisamente complicato da un punto di vista del programma di pdftex, anche perché la ricerca dell'ottimo deve essere eseguita velocemente e con il massimo di efficienza e di efficacia.

27.1 Divisione dei capoversi

Cominciamo a riesaminare il procedimento di divisione dei capoversi in righe. Se ne è già parlato nel capitolo 25, ma ci torniamo sopra per esaminare meglio questo meccanismo.

Cominciamo a ricordare che il capoverso inizialmente costituisce una sola lunga riga e il motore di composizione deve spezzare questa riga in monconi tutti della stessa lunghezza (ovviamente, tranne l'ultimo moncone che può essere più corto). Se è stato invocato il pacchetto `microtype` esiste una complicazione in più, perché la stessa lunghezza può essere raggiunta anche con i meccanismi di protrusione e di leggera deformazione orizzontale dei font che compongono ogni riga di testo, cosicché in prima battuta, prima, cioè, di applicare questi meccanismi raffinati, le righe di testo potrebbero essere considerate di uguale lunghezza entro una certa tolleranza, dell'ordine di grandezza di qualche centesimo della lunghezza da raggiungere; sarà poi compito delle macro del pacchetto `microtype` quello di allungare o restringere orizzontalmente il font presente nella riga aggiustandone la lunghezza per recuperare quei pochi punti percentuali che mancano o che eccedono rispetto alla lunghezza desiderata. In mancanza di questo pacchetto o della sua piena funzionalità, il compito di regolare la lunghezza della riga al valore desiderato spetta a pdftex che vi provvede allungando o accorciando solo la gomma interparola o altra eventuale gomma presente nella riga. Capito questo concetto, lasciamo da parte la microgiustificazione e concentriamoci sul meccanismo standard.

La lunga riga iniziale potrebbe già di per sé contenere delle penalità positive o negative: per esempio, ogni volta che si è usato un segno di legatura mediante la tilde, `~`, che inserisce nel testo uno spazio indivisibile, prima del quale non si può andare a capo, viene inserita una penalità 'infinita', che per TEX è un qualunque valore maggiore o uguale a 10 000. Oppure potrebbe essere stato inserito un comando `\\` che ordina di spezzare la riga in quel punto; e questo 'ordine' viene esplicitato inserendo nella lunga riga una penalità negativa infinita (un valore minore o uguale a −10 000). Inoltre la descrizione della lingua italiana richiede anche l'inserimento di un valore estremamente elevato di demerito se la penultima riga di qualsivoglia capoverso terminasse con una cesura.

pdftex va allora a cercare tutti i possibili punti di divisione in righe dell'intero capoverso, che per ora forma una sola lunga riga, associando ad ogni punto di divisione un valore di demerito. Questo valore di demerito viene determinato in base alle penalità associate al quel punto di divisione, ai demeriti conseguenti alla presenza di una cesura alla fine della riga, alla bruttezza della riga determinata in base all'allungamento o accorciamento della gomma interparola e dell'altra gomma eventualmente inserita dal compositore, per esempio mediante dei *leaders*.

Per calcolare i demeriti associati ad ogni possibile punto di divisione pdftex deve determinare la bruttezza di ogni riga; questa è associata al fatto che la gomma complessiva presente nella riga sia stata allargata o accorciata molto o poco in relazione all'*allungabilità* o all'*accorciabilità* complessiva presente nella riga di cui si sta valutando la possibile fine. Sia z l"allungamento o l'accorciamento

subito dalla gomma per raggiungere la lunghezza desiderata; z può essere positivo (allungamento) o negativo (accorciamento). Sia x l'allungabilità complessiva presente nella riga e sia y la sua accorciabilità. La bruttezza b è data da una espressione numerica di questo tipo:

$$b = 100 \times \begin{cases} \left(\frac{z}{x}\right)^3 & \text{se è avvenuto un allungamento} \\ \left(\frac{|z|}{y}\right)^3 & \text{se è avvento un accorciamento} \end{cases} \tag{27.1}$$

saturata al valore 10 000 se per caso il valore calcolato con l'espressione 27.1 risultasse maggiore.

Va notata l'asimmetria delle espressioni per l'allungamento e per l'accorciamento; è ovvio che l'accorciamento massimo di una riga non può superare l'accorciamento disponibile, quindi la bruttezza associata ad un accorciamento non supera mai il valore 100. Invece l'allungamento può essere maggiore dell'allungabilità, per cui la bruttezza associata ad un allungamento può essere molto grande. Si nota anche che se l'allungabilità o l'accorciabilità sono infinite, la bruttezza è nulla. La presenza della terza potenza fa sì che allungamenti o accorciamenti piccoli rispetto alle rispettive allungabilità o accorciabilità sono del tutto trascurabili, mentre diventano sempre più importanti quando questi rapporti sono vicini all'unità o la superano.

pdftex cerca di evitare che due righe adiacenti abbiano dei valori molto diversi di allungamento o di accorciamento, meno che mai una accorciata e la vicina allungata; considera le righe *decenti* se la loro bruttezza è inferiore al valore 12, mentre se lo eccede, misura la loro *compatibilità visiva* e se due righe adiacenti risultassero incompatibili aggiungerebbe dei demeriti al quel particolare punto di divisione.

In ogni caso calcolata la bruttezza di ogni riga e specificati i demeriti, pdftex deve determinare la bruttezza complessiva delle possibili divisioni in righe di un dato capoverso; per far questa somma calcola i demeriti associati ad ogni possibile punto di divisione e sceglie quella sequenza di punti di divisione che totalizza la minima somma di demeriti.

Intanto pdftex toglie dalla lista tutti i possibili punti di divisione ai quali siano associate penalità p maggiori o uguali a 10 000 e tutti i punti in cui la bruttezza sia superiore alla `\pretolerance` o `\tolerance` correnti; ricordiamo che questi due valori servono per valutare una prima possibile divisione in righe senza cesure, e poi una seconda divisione con le cesure in fin di riga; se la prima non è andata a buon fine, cioè se tutti i possibili punti di divisione alla fine di ogni parola avevano bruttezza superiore al valore di `\pretolerance`, allora procede ad eseguire la seconda divisione con le cesure. In questo testo `\tolerance` vale 200 e `\pretolerance` vale 100.

pdftex deve anche tenere conto della penalità `\linepenalty` legata al numero complessivo di righe in un capoverso; il valore predefinito per questa penalità è $l = 10$, ma il file di classe potrebbe definire un valore maggiore per cercare di

mantenere i capoversi il più corti possibili. Il valore dei demeriti d associati ad ogni possibile punto di divisione, fra quelli rimasti nella lista, viene determinato con l'espressione seguente:

$$d = \begin{cases} (l-b)^2 + p2 & \text{se } 0 \le p < 10\,000 \\ (l+b)^2 - p^2 & \text{se } -10{,}000 < p < 0 \\ (l+b)^2 & \text{se } p \le -10\,000 \end{cases} \tag{27.2}$$

L'espressione per i demeriti presenta una discontinuità che ha i suoi motivi, nei dettagli dei quali non scendiamo. Queste espressioni servono per dare una misura di demerito approssimativamente proporzionale alla somma del quadrato della bruttezza con il quadrato della penalità. Minimizzare la somma dei demeriti per un intero capoverso, vuol dire pressappoco minimizzare la bruttezza di ogni riga. Questo è appunto uno dei punti di forza di questo algoritmo che può venire ottenuto solo con la composizione differita eseguita dai vari motori di composizione del sistema TEX.

Esso è tanto preciso nel determinare il minimo demerito associato ad una particolare divisione in righe, che se un capoverso andasse ritoccato per aggiungere un virgola o una breve congiunzione, potrebbe succedere (e a chi scrive è successo di osservare questo fenomeno) che il capoverso così accresciuto nel numero delle lettere che contiene, può risultare una riga più corto rispetto a prima di eseguire la modifica: non è frequente, ma succede. Questo dipende dalla raffinatezza della ricerca del minimo eseguita dal motore di composizione.

Finito di scomporre il capoverso in righe, le scatole orizzontali che contengono ogni riga vengono accodate alla scatola 255 riservata all'uso di deposito temporaneo del testo composto. La scatola 255 svolge quello che in inglese si chiama *galley proof*, una parte di testo da stampare come bozza, con il testo composto ma non impaginato. Ad ogni capoverso accodato alla scatola 255, viene anche attivato il meccanismo della formazione della pagina; se la scatola 255 contiene abbastanza materiale, la prima parte della scatola viene emessa nel file di uscita e quanto avanza serve per accodarvi il testo che via via viene ulteriormente composto e che verrà usato per creare le pagine successive.

27.2 Divisione della pagine

La divisione delle pagine avviene secondo gli stessi principi seguiti per la divisione dei capoversi, con alcune evidenti varianti. Mentre un capoverso da dividere in righe forma una lista orizzontale di caratteri, penalità, grumi di gomma, eccetera, un testo da dividere in pagine è costituito dalla lista verticale conservata nella scatola 255; questa lista è formata dalle righe già composte per il testo, dalle scatole che contengono, per esempio, singole espressioni matematiche, da penalità, da grumi di gomma verticale, ma non solo. Essa contiene anche dei segnaposti per inserire oggetti che sono stati conservati altrove: gli inserti delle note a piè di pagina, le note marginali, le figure, le tabelle e gli altri oggetti flottanti.

pdftex, nell'estrarre le righe da riversare in una pagina del file di uscita, deve tenere conto dei segnaposti e intercalare alle righe di testo quanto questi segnaposti segnalano. Per le note in calce deve tenere presente della riga dove appare il richiamo in modo da mettere la nota nella stessa pagina dove essa è richiamata; e se non ci riesce deve mettere una forte penalità per questo 'insuccesso'; deve mettere anche una penalità se la nota può cominciare nella pagina ma non ci sta tutta, per cui deve estrarne solo alcune righe per rimettere le righe che avanzano in una posizione utile per la pagina successiva, inserendovi anche una penalità per tenere conto di questo 'insuccesso'. Deve vedere se le note marginali stanno nella pagina o se devono essere spostate nella pagina successiva; deve controllare anche se le note marginali sono abbastanza separate le une dalle altre, ed eventualmente deve spostare i segnaposti per farcele stare, ma aggiungendo altre penalità. Deve esaminare le code degli oggetti flottanti, controllando che stiano sulla pagina, e dove; se non possono stare sulla pagina deve lasciarli in coda, in modo da poterne differire il collocamento ad una pagina successiva; deve tenere conto dei comandi `\clearpage`, e simili, che hanno messo nella lista verticale speciali penalità negative minori di $-10\,000$, e che funzionano da codici per particolari azioni che non richiedono solo un salto di pagina, ma anche lo svuotamento forzato delle code degli oggetti flottanti. Determinata la divisone migliore, che generalmente non supera l'altezza della gabbia del testo, la procedura per la spedizione della pagina completa al file di uscita provvede a inserire la testatina e il piedino.

Come si vede, la formazione delle pagine è sotto molti aspetti più complessa della formazione dei capoversi, ma poi dopo aver deciso che cosa mettere nella pagina, il punto migliore in cui porre la divisione avviene con lo stesso algoritmo usato per i capoversi, nel senso che ora non ci sono due possibili scelte, quella senza cesure e quella con cesure in fin di riga, ma c'è una sola possibile scelta, quella che minimizza i demeriti complessivi della pagina.

Il calcolo della bruttezza avviene più meno nello stesso modo, solo che la gomma interparola è sostituita dalla gomma interlinea `\lineskip`, che però, solitamente, non ha nessuna componente di allungabilità o accorciabilità; della gomma prima della prima riga di ogni capoverso `\parskip`, che generalmente ha una lunghezza naturale nulla e una modesta allungabilità. Della gomma inserita dai comandi di sezionamento prima della loro prima riga, oltre che della penalità inserita dopo la loro fine, in modo da impedire un salto di pagina dopo il titolo (cosa che notoriamente avviene, invece, con i comuni word processor); della gomma presente prima o dopo gli oggetti fuori testo, dalle espressioni matematiche agli oggetti flottanti in genere. La formula della bruttezza è simile all'espressione 27.1 con le ovvie differenze di significato dei simboli presenti.

Invece i demeriti sono determinati diversamente; essi rapprendano il prezzo che bisogna pagare se si interrompesse la pagina nel punto dove lo si calcola; essi sono perciò chiamati *costo* e vengono indicati con il simbolo c; il costo è tanto maggiore quanto peggiore risulta la pagina; per ogni possibile punto di divisione

della pagina il costo associato vale;

$$c = \begin{cases} p & \text{se } b < \infty \text{ e } p \leq -10\,000 \text{ e } q < 10\,000 \\ b + p + q & \text{se } b < 10\,000 \text{ e } |p| < 10\,000 \text{ e } q < 10\,000 \\ 100\,000 & \text{se } b = 10\,000 \text{ e } |p| < 10\,000 \text{ e } q < 10\,000 \\ \infty & \text{se } (b = \infty \text{ oppure } q \geq 10\,000) \text{ e } p < 10\,000 \end{cases} \tag{27.3}$$

dove: b rappresenta la bruttezza, ed è un numero compreso fra 0 e 10 000, o 'infinito' quando la divisione corrisponde ad una pagina che supera l'altezza della gabbia del testo – questo non dovrebbe succedere mai, ma talvolta succede a causa di inserti troppo grandi –; p rappresenta la penalità; q rappresenta le `\insertpenalties`, la somma delle penalità per le note differite o spezzate fra pagine. La divisione finale della pagina avviene nel punto nel quale la somma dei costi associati ai singoli punti di divisione viene minimizzata.

27.3 Cosa fare se...

Come si vede, dunque, l'algoritmo di ottimizzazione della divisione in pagine della scatola 255, con i suoi annessi e connessi, è diverso da quello della divisione dei capoversi, anche se è ispirato da una filosofia simile. Si tratta sempre di scegliere il punto dove i costi sono minimi; se quello è il punto in cui essi sono minimi, significa che ogni altro punto di divisione è peggiore; quindi non c'è da stupirsi se qualche volta pdftex spezza le pagine diversamente da come noi ce lo immagineremmo, se cioè colloca gli oggetti flottanti diversamente da dove ce lo aspetteremmo.

Certo potremmo cambiare i criteri con i quali le parti della pagina possono venire utilizzate per gli oggetti flottanti e per il testo; certo potremmo cambiare le penalità associate alle righe orfane e vedove. E in effetti chi scrive ha spesso usato questi espedienti, ma con grande cautela. I valori preimpostati sono degli ottimi compromessi e cambiarli richiede giudizio e comprensione del meccanismo.

Vale la pena di raccogliere qui alcune informazioni esposte anche altrove.

27.3.1 Collocazione degli oggetti flottanti

pdftex mette gli oggetti flottanti dove lo consentono le condizioni imposte dai vari parametri specificati dalla classe e descritti approfonditamente nel paragrafo 29.10. Come ricordato, quei parametri e quelle frazioni di pagina possono venire cambiati un pochino, ma non troppo. Se si assegna, per esempio, una percentuale del 90% alla frazione di pagina destinata agli oggetti in testa alla pagina, si rischia di avere troppo poco spazio dedicato al testo, e se questo contiene titoli di paragrafi o elementi fuori testo come le espressioni matematiche, questa parte di testo non potrebbe stare nella restante frazione del 10%, e la pagina risulterebbe mozza oppure l'oggetto flottante potrebbe venire spostato ad una pagina successiva.

Se il compositore usa troppo spesso il parametro di collocamento `!` rischia di fare peggio di quanto LaTeX farebbe per conto suo. Si ricorda che questo specificatore non modifica l'osservanza dei parametri solo per la posizione dichiarata con il primo degli specificatori, ma li riguarda tutti. Usare la serie di specificatori `htbp` serve a poco, perché va a finire che se `pdftex` non può usare la posizione specificata con `h`, allora l'oggetto mobile non può più essere collocato "lì" e rischia di restare in coda a lungo; eventualmente la coda degli oggetti flottati può venire svuotata d'autorità o alla fine di un capitolo o per effetto di un comando `\clerapage` (eventualmente passato come argomento a un comando `\afterpage`). È meglio evitare di specificare `h`; non è un caso che le posizioni predefinite in LaTeX sono proprio `tbp`, perché se esso non può andare in testa a 'questa' pagina, vi può andare in calce, oppure può andare in testa o in calce in una pagina successiva, oppure quando vengano svuotate le code. Se l'oggetto è di grandi dimensioni, è meglio specificare solo `p`, cosicché alla prima pagina utile esso viene collocato in una pagina a sé stante (sempre che la coda non sia già intasata) e spedito al file di uscita.

Raramente capita che le code intasate restino tali via via che si aggiungono altri elementi in coda; quando si supera il numero di 18 oggetti in coda, i successivi vengono persi; sì, è vero, il programma emette un avviso, ma è una magra consolazione. Se si ha anche solo il dubbio che la coda possa essere intasata da un oggetto troppo grande per condividere una pagina con del testo, è meglio verificare la cosa durante la lavorazione del documento emettendo un comando `\clearpage`, così da vedere che cosa ci sia in coda, e poi eventualmente cambiare i descrittori di posizione, emettere uno o più `\clearpage` passandoli al programma come argomenti di `\afterpage`; insomma in modo da superare le barriere messe dal programma contro una brutta composizione tipografica. Agendo con giudizio, si può aggiustare a mano quanto il programma non riesce a gestire da solo; ma questo va fatto in parte durante la lavorazione del documento e in parte alla fine per dare i tocchi finali al lavoro compiuto.

27.3.2 Le equazioni ingombranti

Le 'piccole' equazioni fuori testo, fra espressione matematica vera e propria e spazi sopra e sotto, impegnano l'equivalente di tre righe di testo; espressioni matematiche più complesse possono richiedere un numero di righe molto maggiore. Se una di queste espressioni, 'piccole' o 'grandi' che siano, compare in testa ad una pagina, è molto probabile che la pagina precedente sia stata dichiarata 'Underfull \vbox' da `pdftex`, perché ha dovuto stiracchiare troppo la gomma disponibile sulla pagina per non farla sembrare mozza; in realtà non è mozza, ma è brutta a causa dell'eccessivo allargamento dei contrografismi che separano le varie parti della pagina.

In questi casi non ci sono cure diverse da quelle di:

1. modificare il testo della pagina stiracchiata, in modo che contenga *più* testo;

2. modificare il testo della pagina stiracchiata, in modo che contenga *meno* testo;

3. allungare la pagina stiracchiata usando il comando `\enlargethispage` con o senza asterisco, in modo che lo specchio di stampa sia un poco più grande così da contenere una o due righe in più;

4. accorciare lo specchio di stampa della pagina stiracchiata, sempre usando `\enlargethispage` ma usando un allungamento negativo, così da richiedere uno stiracchiamento minore;

5. usare il comando primitivo `\looseness` con l'argomento positivo o negativo per allungare o accorciare, se possibile, capoversi scelti della pagina stiracchiata così da riempire meglio la pagina; 'capoversi scelti' vuol dire capoversi sufficientemente lunghi da avere molto spazio interparola su cui giocare per ottenere l'effetto desiderato. Il numero che si assegna al comando `\looseness` indica di quante righe allungare o accorciare un capoverso; a meno che non si tratti di un capoverso molto lungo, è difficile allungare o accorciare un capoverso di più di una riga. Tuttavia spesso uno o due capoversi allungati di una riga possono aggiustare una pagina in modo difficilmente riconoscibile a occhio nudo.

È chiaro che questi provvedimenti, tranne quelli che prevedono l'aumento o la diminuzione del testo, sono provvedimenti da prendere solo nella fase finale di messa a punto dell'impaginazione, quando il testo è praticamente finito di comporre, altrimenti quegli aggiustamenti finirebbero per dover essere rivisti ripetutamente ad ogni modifica del testo che li precede.

Queste cose vanno viste con molta cura anche quando si eseguono degli inserti non flottanti, come una piccola figura contornata dal testo; questa è legata al particolare capoverso il cui testo le gira attorno e se la modifica del testo che lo precede lo sposta più in alto o più in basso, la figurina fissa rischia di andare a finire dove l'algoritmo di formazione delle pagine inserisce un salto pagina; la figurina potrebbe rimanere lì, penzoloni, alla fine della pagina, a fianco di un paio di righe del suo capoverso, che però proseguirebbe nella pagina successiva con la rientranza vuota inizialmente riservata per la figurina. Cose queste che sconsigliano fortemente di fare uso di piccole figure contornate dal testo, se non nel *primo* capoverso di un capitolo, perché non vi è nessun testo prima che possa spostare il primo capoverso più in basso.

27.3.3 I sezionamenti

Anche i titoli dei paragrafi e delle sezioni di rango successivo possono dare problemi. Si è detto e ripetuto che lo spazio verticale che segue un titolino è uno spazio 'indivisibile', uno spazio verticale *prima e dopo* del quale non può avere luogo un salto di pagina. quindi il titolo del paragrafo è distanziato dalla prima riga del suo primo capoverso, ma non ne può venire staccato. Se

avvenisse un salto di pagina dopo la prima riga del capoverso, essa rimarrebbe orfana; per evitare questo la classe definisce una penalità `\clubpenalty`, insieme alla sua compagna `\@clubpenalty`, che viene inserita dopo la prima riga di ogni capoverso, proprio per evitare le righe orfane; dunque, sebbene non sia strettamente impossibile (con le penalità predefinite) un salto di pagina dopo la prima riga, la seconda riga appare subito sotto alla prima riga; ma se il capoverso ha ancora solo una terza riga, se avvenisse un salto di pagina dopo la seconda riga, la terza riga resterebbe vedova; una forte penalità `\widowpenalty` viene inserita prima dell'ultima riga di ogni capoverso, proprio per evitare la formazione di righe vedove, e ci riesce quasi sempre con i valori predefiniti. Ma tutto ciò implica che la riga del titolino, seguita dal suo spazio e seguito ancora da almeno due righe di testo costituisce un insieme 'monolitico' di grandi dimensioni che o sta tutto in fondo alla pagina o viene spostato tutto all'inizio della pagina successiva (naturalmente se è a metà pagina il problema non esiste). I valori di `\clubpenalty` e di `\widowpenalty` sono generalmente fissati dai file di classe, o dai file di descrizione della lingua, ma il compositore può cambiarne i valori in modo accorto; i valori predefiniti di solito sono piuttosto severi, ma generalmente non sono valori infiniti: in questo testo `\clubpenalty` vale 10000, e `\widowpenalty` vale 3000.

Si noti che sono considerate orfane e vedove anche la prima riga dopo una equazione fuori testo o l'ultima riga prima di una equazione fuori testo; quindi il problema delle righe orfane e vedove si pone anche con le equazioni fuori testo.

In questi casi si può agire come negli altri casi, allungando o accorciano il testo che precede il blocco 'monolitico'; volendo, trattandosi di un sezionamento si potrebbe inserire subito prima del comando di sezionamento un comando `\newpage`; oppure talvolta è preferibile un comando `\goodbreak`; è vero che se il comando agisce, lascia una pagina mozza, ma se si corregge il testo precedente questo comando potrebbe restare inefficace, grazie alla piccola penalità che esso inserisce nella lista verticale. Il comando di basso livello (primitivo) `\filbreak` è troppo forte e in caso di modifiche agirebbe anche lontano dal fondo della pagina. Per questo testo è stato definito un comando `\goodpagebreak`, con il suo alias `\buonfinepagina`, che mozza la pagina corrente nel punto in cui appare il comando se al completamento della pagina mancano meno di un numero specificato di righe; di default questo numero vale 4.

27.4 Conclusioni

La casistica potrebbe allungarsi moltissimo. Però ogni qual volta ci si trovi con una pagina mal formata, con troppo spazio bianco, la conoscenza del meccanismo di divisione delle pagine e il gioco sottile dell'allungamento della gomma in competizione con le penalità, e con le dimensioni degli oggetti di grandi dimensioni, permette di risalire alle cause e ad apportare le modifiche del caso per risistemare la pagina mal riuscita.

Capitolo 28

Trattamento degli errori

Quando il programma di composizione incontra nel file sorgente delle parti che non riesce ad interpretare o non riesce ad eseguire, non sa cosa fare e si ferma con un messaggio d'errore; spesso il messaggio di errore che accompagna la descrizione dove ha trovato la difficoltà è sufficientemente chiaro. Queste in generale sono situazioni semplici, altre volte gli errori sono molto subdoli e i messaggi sono troppo vaghi.

28.1 Errori ortografici nei nomi dei comandi

La situazione elementare, quella di aver scritto in modo errato un comando, è già stata descritta alla fine del capitolo 5, e non ci si torna sopra. Quando il lettore arriva a leggere questo capitolo estremamente tecnico, vuol dire che si è scontrato con un errore davvero grave, ma ha già acquisito abbastanza esperienza per affrontare le situazioni elementari.

28.2 Errore nella ricerca dei file

Nello stesso modo si risolvono errori del tipo "File not found"; accanto al messaggio appare il nome del file che il programma di composizione non avrebbe trovato. L'obiezione più frequente che ci viene in mente è: "Ma come?! Se il file è nella cartella di lavoro!". Prima di lasciarci andare alla irritazione del caso, conviene leggere con calma e attenzione il nome del file che il programma non ha trovato e ragionarci sopra.

1. Il nome del file potrebbe contenere degli spazi; è possibile far leggere al programma nomi di file che contengano degli spazi, ma la la soluzione, oltre ad essere "system dependent" e a eliminare ogni possibilità di trasportare il file su un'altra macchina, è da evitare in modo assoluto; la soluzione vera è quella di togliere gli spazi dai nomi dei file. Se si ritiene che i nomi dei

file senza spazi siano illeggibili, si può sempre riempire lo spazio con un trattino (il tratto ribassato è sconsigliabile), per esempio da `My file.tex` a `My-file.tex`, dove la leggibilità è certamente conservata ma lo spazio non c'è più.

2. Il file col nome senza spazi è effettivamente lì, ma il sistema operativo distingue le lettere maiuscole da quelle minuscole nei nomi dei file; tutti i sistemi UNIX hanno questa particolarità, solo i sistemi operativi di tipo Windows non fanno differenza fra `Myfile.tex` e `myfile.tex`. La soluzione è controllare bene che si sia chiesto al programma di leggere il file il cui nome sia scritto con la stessa ortografia di come è registrato sul disco.

3. L'estensione `.tex` *non deve mai essere specificata*; i programmi del sistema TEX appendono sempre l'estensione `.tex` ai nomi di file che ne siano privi, ma a quel punto è bene essere sicuri di non aver inserito spazi spuri, cioè è importante che scriviamo `\input{myfile}` e non `\input{␣myfile␣}`.

4. Anche i file grafici è bene che siano specificati senza estensione; non è obbligatorio, ma è conveniente. Però, per esempio, stabilito che pdflatex non legge i file con estensione `.gif`, non è possibile cambiare l'estensione `.gif` in `.png`; l'estensione non è una decorazione inutile di un file, ma dice che formato ha il suo contenuto; cambiare l'estensione non cambia il contenuto. Eppure questo è un tipico errore che viene commesso più spesso di quanto non si creda, e a cui il programma reagisce con un messaggio di "File not found".

5. Il file potrebbe essere protetto e accessibile solo al proprietario del file o all'amministratore della macchina; prima di poterlo aprire bisogna modificarne gli attributi per consentirne l'apertura.

6. Un file di classe, o un pacchetto, o un qualunque altro file di servizio potrebbe essere stato installato nella cartella giusta, ma è possibile aver dimenticato di aggiornare il database dei nomi dei file; il programma di composizione non può trovarlo in quanto non è elencato i quei database. Esiste un database dei nomi dei file per ogni albero in cui è suddiviso l'insieme dei file di lavoro del sistema TEX; generalmente i file delle distribuzioni ufficiali sono messi in alberi accessibili solo all'amministratore della macchina. Nelle macchine Windows questo concetto di utente privilegiato ad eseguire qualunque operazione esiste, ma è meno preciso di come è definito sulle macchine UNIX. È quindi possibile che l'utente pensi di fare cosa giusta ad aggiungere i propri file personali nelle stesse cartelle dove risiedono i file del sistema. Questo invece sarebbe un grave errore, perché al primo aggiornamento dei file di sistema i file personali sparirebbero. I file personali vanno inseriti in una struttura ad albero conforme al TDS (*TEX Directory System*) ma la radice di questo albero deve essere in una zona personale; sulle macchine Linux, questa zona è la cartella `Home`, indicata con `~/`, e ad

ogni modifica del contenuto dell'albero personale bisogna aggiornare il database dei nomi dei file in esso contenuti usando gli strumenti specifici delle distribuzioni del sistema TEX su quelle macchine. Nelle macchine Windows la cartella corrispondente a `Home` non è ben definita, sebbene le versioni più recenti dei sistemi operativi creati da Microsoft abbiano una sorta di cartella personale, generalmente radicata in `Documenti\`⟨*NomeUtente*⟩`\`. Questa struttura di albero personale, va elencata fra gli alberi di MiKTEX, in modo che quando si apre la cartella `MiKTeX Settings` il tasto `Refresh databases` possa agire anche sull'albero personale. Sulle macchine Mac, la radice dell'albero personale è in `~/Library/` e non ha bisogno di venire aggiornata; MacTEX lo sa e ne sfrutta le funzionalità.

Come si vede sono tutte soluzioni semplicissime che non spaventano nessuno che conosca bene la propria macchina e il proprio sistema operativo. È vero; qualche volta situazioni di questo genere mettono in imbarazzo, ma la soluzione è poi immediata.

28.3 Ciclo infinito

L'unico errore per il quale ogni programma del sistema TEX non emette messaggi è forse l'errore più grave, l'entrata in un ciclo infinito; ogni programma, classe, o pacchetto del sistema TEX è in grado di eseguire delle chiamate ricorsive, ciò che avviene quando una macro direttamente o indirettamente richiama se stessa; perché questa ricorsione abbia termine è necessario che lungo questa operazione ciclica qualcosa cambi e un test ne verifichi il cambiamento al fine di non ripercorrere il ciclo all'infinito. Se ciò non accade il programma continua a riciclare all'infinito; a parte il tempo perso a riciclare a vuoto, può succedere che il programma di composizione scriva qualcosa nel file di uscita o nei file ausiliari. Se ciò malauguratamente succedesse e l'operatore non fosse pronto a "uccidere" il programma, si finirebbe per saturare l'intero disco con danni non facilmente quantificabili, ma sicuramente danni seri al contenuto del disco e alle operazioni necessarie al suo ripristino.

Questo genere di cose possono succedere quando si scrivono delle macro ricorsive; generalmente queste macro servono per eseguire elaborazioni che non coinvolgono l'utente finale, ma interessano moltissimo a coloro che scrivono classi oppure pacchetti o anche soltanto alcune macro per il preambolo del proprio documento.

L'unica via per uscirne è essere molto tempestivi, prima che possano essere danneggiate le informazioni sul proprio hard disk; spesso l'editor dispone di un pulsante di emergenza; in TEXShop c'è un bottone Abort; in TeXworks il bottone verde per lanciare la compilazione, può essere cliccato e oltre a diventare rosso arresta il programma. Sugli editor specifici per le macchine Linux o ci sono analoghi bottoni, oppure basta eseguire le scorciatoie di tastiera come Ctrl + Y, o simili. Sulle macchine Windows la procedura è più elaborata e più lunga,

in quanto bisogna arrestare il programma agendo sul Task Manager, che però solitamente non ha un bottone sempre disponile in una delle barre, e quindi si consuma del tempo prezioso per eseguire tutta la procedura di "uccisione" del programma entrato in un ciclo infinito. Con le macchine con un sistema operativo ti tipo UNIX (Linux e Mac) si può sempre agire dalla finestra comandi dando prima il comando `ps ax` per determinare il numero del processo e poi dare il comando `kill 8902` (se il numero del processo è stato rilevato essere 8902). Le macchine Mac hanno anche il menù "mela", in alto a sinistra nella barra superiore, con la voce "Force quit"; cliccando su questa voce si apre una piccola finestra di dialogo dove sono elencati i processi dell'utente che sono in esecuzione, e di lì è facile individuare quella componente di TEX che sta facendo capricci; la si può selezionare e "uccidere".

28.4 Gruppi aperti

Un altro errore molto infido è la mancanza della parentesi chiusa di un gruppo; talvolta questa parentesi mancante provoca solo un avviso scritto verso la fine del file `.log` che spiega che il programma è terminato mentre si era dentro un gruppo; il messaggio spiega anche a quale livello di annidamento si trovava quel gruppo; qualche volta riesce a dire anche in quale riga di quale file quel gruppo era stato aperto. Talvolta il gruppo era stato aperto dentro un file incluso nel flusso di elaborazione mediante il comando `\include`; in questo caso si accorge che il gruppo non è chiuso quando ha finito di leggere il file incluso, ma il messaggio non riesce a dire altro che c'è un gruppo non chiuso e indica solo la riga del master file dove c'è il comando `\include`. È una informazione minima, ma almeno si riesce a sapere quale file il programma stava leggendo quando è arrivato alla fine con il gruppo ancora aperto.

Questo tipo di errore è relativamente facile da scovare; la maggior parte degli *shell editor* sono in grado di riconoscere le graffe appaiate; alcuni, in base a queste informazioni riescono a disegnare a lato della schermata dell'editor una struttura di linee che descrive graficamente la struttura del file e dei gruppi che contiene, siano essi ambienti delimitati dai comandi `\begin` e `\end`, siano essi gruppi delimitati dalle graffe. Chi scrive trova più comodo il funzionamento di TEXShop, che non ha la barra laterale; in questo *shell editor*, e in diversi altri per altri sistemi operativi, cliccando su una graffa l'editor cerca la sua compagna ed evidenzia tutto il gruppo con un colore di sfondo diverso; sia che il gruppo sia breve, sia che coinvolga diverse schermate del file sorgente, il brusco cambiamento di colore indica subito l'esistenza di un gruppo; al massimo potrebbe essere stato chiuso nel punto sbagliato, ma almeno il gruppo esiste. Quando invece il gruppo non è chiuso, non succede nient'altro e l'errore è trovato, basta solo andare a cercare il punto giusto per porre la parentesi chiusa mancante.

L'errore infido della mancata chiusura di un gruppo può talvolta essere molto difficile da scovare in un lungo documento non suddiviso in file da includere; anche in un documento suddiviso in diversi file inclusi può essere particolarmente

dannoso se la mancanza del gruppo provoca messaggi d'errore incomprensibili e che non si riesce a mettere in relazione a quello che si crede di avere scritto dentro il file sorgente. In questo caso si può procedere col metodo della dicotomia; si sposta la schermata dell'editor a metà del file (unico o incluso), non dentro un ambiente, ma prima o dopo l'apertura o, rispettivamente, la chiusura di un ambiente, e lì si inserisce una riga contenente solo

```
\end{document}
```

A questo punto si lancia il programma di composizione; se l'errore continua a presentarsi, la mancanza della chiusura del gruppo è nella prima metà del file; altrimenti è nella seconda metà. A questo punto si toglie la precedente specificazione di fine documento, lasciando al suo posto un contrassegno di posizione: potrebbe essere quella riga stessa dopo averla commentata. Poi la si inserisce nuovamente a metà della parte incriminata e si ripete il processo quante volte occorre fino a restare con una parte di file di poche decine di righe dove dovrebbe essere più facile trovare che cosa manca.

Si tratta di soluzioni estreme? Certo, quando non c'è nessuna informazione utile per capire la natura dell'errore bisogna ricorrere a soluzioni estreme; ammetto di averle dovute usare molto raramente ma in quasi trent'anni d'uso di LaTeX potranno essere state una mezza dozzina di volte. Certo mi capitava più facilmente quando la distribuzione del sistema stava su un paio di floppy da $5\frac{1}{4}''$ e la documentazione era virtualmente inesistente; ora sono anni che non mi capita più, grazie sia alla migliore diagnostica e all'esperienza accumulata.

28.5 Mancata apertura di un gruppo

Manca sempre una graffa, questa volta aperta, mentre è presente la graffa chiusa. Il metodo di cliccare su graffe aperte o chiuse e controllare quale parte del file sorgente viene evidenziata funziona sempre, ma prima bisogna capire che si tratta di una graffa aperta mancante. Spesso il messaggio d'errore fa riferimento ad una graffa chiusa ed informa l'utente che:

```
./errori.tex:55: Too many }'s.
l.55 ... graffa chiusa}
                        ed informa l'utente che:
?
```

Si capisce subito di che cosa si tratta e che l'errore non è grave, per cui si può tranquillamente lasciar continuare la compilazione fino in fondo; bisogna però ricordarsi che la riga 55 del file `errori.tex` (come dice il messaggio) contiene una graffa chiusa priva di una precedente compagna graffa aperta e l'erroretto va comunque corretto.

28.6 Interferenza con le definizioni

Talvolta la graffa aperta mancante cade in una posizione insolita dove la corrispondente graffa chiusa ha un senso per il programma interprete. La graffa chiusa non viene rilevata come erronea, e l'interpretazione continua fino ad un punto dove il programma non capisce più niente ed emette un messaggio d'errore che non ha nulla a che fare con il punto dove manca effettivamente la graffa aperta. Si consideri il codice seguente:

```
...
\newcommand\mycommand{%
\ifnum ....
    \def\RCi{\relax}\let\RCuno\RCi
    \def\RCii\relax}\let\RCdue\RCii
    \def\RCiii{\relax}\let\RCtre\RCiii
    ...
\else
    ...
\fi
    ...
}
```

Viene usato un test con la sintassi primitiva di TEX; la mancanza della graffa aperta nella definizione di `\RCCii` fa sì che il test iniziato con `\ifnum` non si chiuda all'interno della definizione. Se, usando TEXShop, avessimo cliccato sulla parentesi che apre la definizione di `\mycommand` avremmo trovato che questo "gruppo" si chiude dopo il comando `\relax` che segue la definizione di `\RCii` e ci saremmo accorti subito dell'errore. Non l'abbiamo fatto e abbiamo mandato in compilazione il documento contenente queste definizioni. Poiché le macro contenute dentro le definizioni "normali" non vengono sviluppate (eseguite), il programma di composizione ci avrebbe avvisato della presenza di un "extra `\else`"; se avessimo forzato il programma a proseguire, avremmo avuto un altro avviso di errore per la presenza di un "extra `\fi`"; avremmo avuto due segnalazioni di errore causati dalla mancanza di una graffa aperta, ma i due messaggi d'errore avrebbero portato la nostra attenzione sui comandi condizionali, non sulle graffe; se avessimo forzato ancora il programma a proseguire, finalmente esso ci avrebbe avvisato della presenza di "Too many }'s" e finalmente avremmo capito che mancava una graffa aperta o che c'era una graffa chiusa di troppo forzandoci a rivedere le graffe e a trovare finalmente la mancanza della graffa aperta.

28.7 File personali che provocano conflitti

In un file di classe avevo scritto il seguente codice:

```
\AfterEndPreamble{%
```

```
   \@ifundefined{virgola}{%
   \DeclareMathSymbol{\virgola}{\mathpunct}{letters}{"3B}
   \DeclareMathSymbol{\virgoladecimale}{\mathord}{letters}{"3B}}{}
}
```

Volevo evitare di ridefinire alcune assegnazioni eventualmente eseguite in altri file letti nel preambolo. Il lettore può riconoscere parte del codice illustrato nel capitolo 19 a proposito della virgola intelligente. Quel tipo di codice appare già nella mia versione personale del file di descrizione della lingua italiana `italian.ldf` letto da *babel* quando specifico la lingua italiana. Il mio errore consiste proprio nell'aver modificato a titolo personale un file avente lo stesso nome del file del sistema TEX che non deve mai essere modificato, mai e poi mai, per nessun motivo. "Mal me ne incolse."

Infatti la documentazione del pacchetto *etoolbox*, col quale è definito il comando `\AfterEndPreamble`, specifica chiaramente che i comandi contenuti nell'argomento di questo comando vengono sviluppati dopo che che è cessato il preambolo (come d'altronde il nome dice con grande chiarezza) e che quindi questi comandi non possono contenere altri comandi usabili solo nel preambolo. Il comando `\DeclareMathSymbol` è appunto uno di questi comandi a validità limitata che possono essere usati solo nel preambolo.

Anche se avevo recepito questo avvertimento, non l'avevo evidentemente interiorizzato e avevo scritto il codice suddetto che funzionava perfettamente e non ricevevo nessun messaggio d'errore. Tranquillo della bontà del codice, avevo mandato il file ad un amico; al quale la cosa non funzionava e riceveva il messaggio d'errore che:

```
... \DeclareMathSymbol
                       may be used only in the preamble
... See the LaTeX Companion for explanation
...
?
```

L'amico mi scriveva dicendomi "ricevo questo messaggio d'errore, ti mando il file `.log`...", e io gli rispondevo "Non riesco a replicare il tuo errore; dammi più dettagli...". Insomma fra noi c'era un dialogo fra sordi, ma il sordo ero io che avevo completamente rimosso l'avvertimento del pacchetto *etoolbox* e non lo collegavo con il messaggio piuttosto esplicito dell'amico.

Il problema ere causato dal mio file personalizzato `italian.ldf` che definiva il comando `\virgola` per cui il codice incriminato veniva saltato grazie al test `\@ifundefined`; ho fatto perdere al mio amico una buona settimana di botta e risposta inconcludenti per colpa mia; finché finalmente mi si è accesa la luce; ho capito il problema e ho potuto rimediare con mille scuse. Per l'ennesima volta ho commesso il "peccato" di ridefinire un file di sistema e per l'ennesima volta ne ho pagate le conseguenze. Ecco perché, tra l'altro, insisto sempre sulla necessità di non commettere un simile errore. Qui almeno aggiungo la raccomandazione di non essere ottusi, come lo sono stato io, a non cogliere il significato vero dei messaggi d'errore.

28.8 Tracciare l'operato del programma

Il sistema TEX non è così rozzo come potrebbe apparire da quanto descritto nei paragrafi precedenti di questo capitolo; quei paragrafi descrivono situazioni estreme alle quali un linguaggio interpretato, come il linguaggio TEX, talvolta non può far fronte.

TEX invece è dotato di un certo numero di comandi primitivi, alcuni dei quali richiedono degli argomenti e tra i pacchetti LATEX ce ne è uno che svolge questo compito molto bene, anche se i puristi preferiscono gestire individualmente i singoli parametri di tracciamento.

28.8.1 I comandi primitivi di tracciamento

I parametri di tracciamento sono descritti qui di seguito; va notato che sono quantità a cui bisogna assegnare un valore positivo; quando si assegna il valore zero diventano inefficaci: la sintassi per tutti è del tipo seguente:

\⟨*comando di tracciamento*⟩ = ⟨*numero*⟩

`\tracingonline` serve per emettere le informazioni di tracciamento sulla console; lo si può usare anche quando si lancia il programma di composizione attraverso uno *shell editor*, ma sarebbe meglio evitarlo specialmente per tracciati lunghi; la memoria della console è relativamente limitata e si rischia di perdere alcune parti del tracciato.

`\tracingmacros` con un valore positivo, meglio se pari a 2, mostra lo sviluppo delle macro via via che queste vengono sviluppate.

`\tracingstats` con un valore positivo mostra l'utilizzazione delle memorie del sistema TEX.

`\tracingparagraphs` serve per mostrare i calcoli eseguiti quando il programma compone un capoverso spezzandolo in righe possibilmente di uguale lunghezza.

`\tracingpages` serve per mostrare i calcoli eseguiti quando il programma spezza la scatola 255 in pagine inserendovi dentro le note e gli oggetti mobili.

`\tracingoutput` serve per mostrare le operazioni che il programma esegue quando deve spedire una pagina composta al file di uscita.

`\tracinglostchars` serve per mostrare quello che succede quando si invoca l'uso di un particolare carattere (glifo) ma il font non lo contiene.

`\tracingcommands` serve per mostrare quello che il programma di composizione fa quando deve eseguire un comando.

`\tracingassigns` mostra quando viene assegnato un valore ad un registro di conteggio di numeri interi, di lunghezze, di lunghezze elastiche, eccetera.

`\tracingrestores` serve per mostrare che cosa viene ripristinato quando il programma di composizione esce da un gruppo.

`\showboxbreadth` indica il massimo numero di oggetti da mostrare in ciascun livello di inscatolamento durante il tracciato.

`\showboxdepth` mostra il massimo livello di inscatolamento da mostrare nel tracciato.

Chi scrive spesso si è servito della coppia di specificazioni:

```
\tracingcommands=2\tracingmacros=2\relax
⟨righe del file sorgente da tracciare⟩
\tracingcommands=0\tracingmacros=0\relax
```

Il difetto è che l'informazione è molto, forse troppo, abbondante; in particolare quando vengono selezionati dei font, quando si compone della matematica, quando si traccia la chiusura di un documento, quando le righe da tracciare coinvolgono un fine pagina e la spedizione di una pagina all'uscita.

Qualunque esempio, anche minimo che si potrebbe fare per l'uso di questi comandi impiegherebbe diverse pagine e non si ritiene che sia di nessuna utilità senza un commento riga per riga, che risulta in realtà superfluo visto che ogni riga del tracciamento è autoesplicativa. Se esiste qualche difficoltà, questa sta nell'interpretazione delle informazioni e nell'abilità del lettore di saltare le parti che non interessano. Tuttavia si impara; la lettura del tracciamento non è spiegata da nessuna parte, ma tutti quelli che hanno eseguito e letto dei tracciati li hanno compresi, certo con pazienza e con molta determinazione, ma ci sono riusciti.

28.8.2 Il pacchetto *trace*

Il pacchetto *trace* sembra poco flessibile, ma sostanzialmente traccia tutto quello che può essere tracciato tranne la selezione dei font, la costruzione delle parti matematiche, la costruzione dei capoversi e la costruzione delle pagine, delle quali espone solo la riga finale. Se si usano male i suoi comandi potrebbe succedere che venga tracciato anche l'operato della routine di uscita, quella che emette nel file di uscita una pagina definitivamente composta con il testo, le note al piede e le note marginali, le eventuali figure e quant'altro venga eseguito durante l'accodamento di una pagina completata al file di uscita.

I comandi per attivare il tracciamento sono molto prevedibilmente `\traceon` e `\traceoff`; essi vanno messi a cavallo della parte che si vuole tracciare e che produce messaggi d'errore; per coloro che creano classi, pacchetti o macro personali, possono servire per scoprire perché le macro accuratamente programmate e realizzate non fanno quello che si vorrebbe che facessero. Se si dovesse mettere `\traceoff` proprio appena prima della chiusura di un gruppo, lo si può omettere, perché il comando `\traceon` inserito dentro lo stesso gruppo cessa il suo effetto alla chiusura del gruppo stesso.

In effetti `\traceon` serve per impostare i parametri di tracciamento in questo modo:

```
\@tracingtrue
\tracingstats=2
\tracingpages=1
\tracinglostchars=1
\tracingparagraphs=1
\errorcontextlines=\maxdimen
\tracingoutput=1
\showboxbreadth=\maxdimen
\showboxdepth=\maxdimen
\errorstopmode
\tracingmacros=2
\tracinggroups=1
\tracingrestores=1
\tracingcommands=2
\tracingassigns=2
\tracingonline\tracingonline@p
```

Il comando opposto `\traceoff` rimette a zero tutti quei parametri. Si noti che `\maxdimen` rappresenta la massima lunghezza trattabile dai programmi del sistema TEX; visto in termini di "scaled points", se si assegna questo valore ad un contatore intero, indica il massimo numero intero (diviso per quattro) gestibile dai programmi del sistema TEX[1]; di fatto assegnare questo valore a `\showboxbreadth` e a `\showboxdepth` equivale ad assegnare un valore talmente grande da poterlo considerare "infinito". `\@tracingtrue` impone il valore `true` alla variabile booleana `@tracing`. `\tracinngonline@p` è una macro che può contenere il valore 0 o 1 a seconda delle opzioni con cui viene chiamato il pacchetto *trace*.

Infine `\errorstopmode` imposta il modo di funzionamento del programma in modo che si arresti ad ogni errore; spesso i programmi di composizione vengono lanciati con altre opzioni, per esempio `nonstopmode` che possono risultare più veloci nell'esecuzione; chi scrive preferisce sempre lavorare in modalità `errorstopmode`, cosi può rendersi conto subito degli errori commessi; avendo disponibile la schermata dello *shell editor* può nella maggior parte dei casi spostarsi alla riga segnalata come erronea, correggere subito quella riga nel file sorgente; contemporaneamente, alla richiesta del programma di composizione, egli introduce l'informazione corretta servendosi della lettera di codice i, molto comoda, appunto, per correggere subito certi errori di ortografia nei comandi, ma tale da poter proseguire subito, generalmente fino alla fine della compilazione dell'intero file.

Questi due comandi, oltre ad "accendere" il tracciamento e a "spegnerlo", fanno sì che le informazioni del tracciamento vengano scritte nel file `.log`.

[1] Il registro `\maxdimen` vale 16383.99998pt; assegnando `\maxdimen` a un contatore intero, questo vale 1073741823, più di un miliardo.

Terminata l'esecuzione del programma di composizione, si può aprire anche con lo stesso *shell editor* con cui si compone il file sorgente dei documenti, quel file `.log` e leggere tutto quello che ha fatto il programma nella parte tracciata.

Di ogni comando che viene eseguito viene mostrata la definizione e quali argomenti vengono passati al comando; poi uno per uno vengono elencati i comandi primitivi oppure vengono "espanse" le macro e viene mostrato su quali argomenti lavorano; si procede così fino alla fine della parte da tracciare, senza mostrare se non qualche breve parte relativa ai cambiamenti di font, quel tanto che basta per sapere quale comando sia stato eseguito e quale font in effetti sia stato selezionato; vengono mostrate tutte le aperture di gruppi con i loro livelli e le corrispondenti chiusure; come vengono ripristinate le macro o i contatori che abbiano ricevuto altre definizioni o altri valori all'interno di ogni gruppo; quali valori avessero le macro oggetto di test di vario genere; quali siano gli esiti dei test e quale biforcazione si sia presa alla fine del test; che cosa indicassero i token impliciti; che cosa contenessero le macro oggetto dei comandi `\expandafter`. Insomma per filo e per segno ogni possibile azione eseguita dal programma di composizione, salvo i dettagli meno interessanti, come per esempio la selezione dei font e poche altre cose; se interessassero queste informazioni si possono usare i comandi di tracciamento descritti nel paragrafo precedente.

La descrizione di queste cose è molto minuziosa; il file `.log` rischia di essere molto, molto grande; la sua lettura molto faticosa; la sua comprensione molto difficile finché non si è presa la mano. È una operazione da cuori molto forti. Ma anche con i comandi primitivi si hanno situazioni dello stesso genere, talvolta ancora più dettagliate.

Mettere qui un esempio vorrebbe dire scrivere un altro libro. Vi rinuncio. Tuttavia non esistono altri metodi per imparare a comprendere i tracciato contenuto nel file `.log` se non quello di provarci e di riprovarci; dopo un paio di volte si fa luce nella nostra visione dell'operato del programma di composizione e la lettura del tracciato non è più così difficile come le prime volte: lungo sì, noioso sì, ma oscuro non più.

Vanno usate alcune piccole astuzie: bisogna stare attenti a non inserire le due macro di accensione e spegnimento dentro una macro che viene usata centinaia di volte ma mettere i due comandi suddetti a cavallo di un breve brano di file sorgente che contenga una sola volta l'esecuzione di quella macro. Bisogna evitare tratti di testo sorgente che contengano numerosi cambiamenti di font; anche se per questi viene presentato solo un riassunto, se i cambiamenti sono numerosi, i riassunti sono numerosi e alla fine si perde di vista quello che interessa. Bisogna ovviamente, trovato l'errore o la causa di un malfunzionamento di una macro, commentare o cancellare quei due comandi, altrimenti il tracciamento continua ad essere eseguito rendendo difficoltosa la lettura di un eventuale altro tracciamento in un altro punto del file sorgente. Insomma si impara ad usare correttamente la funzionalità di tracciamento e a leggerne soltanto le parti che interessano. Diventa un aiuto prezioso, specialmente se si scrivono macro.

28.9 Costruzione dei capoversi e delle pagine

Il tracciamento della costruzione dei capoversi e delle pagine è richiesto quando non si capisce bene perché un capoverso venga composto male o diversamente da come ce lo aspetteremmo; lo stesso vale per la costruzione delle pagine da spedire in uscita.

A differenza degli altri comandi di tracciamento descritti nel paragrafo precedente, questi comandi sono descritti approfonditamente nel TEXbook. I comandi in questione sono `\tracingparagraphs=1` e `\tracingparagraphs=0` per accendere e spegnere il tracciamento della costruzione dei capoversi al livello di dettaglio 1; analogamente per il tracciamento della costruzione delle pagine `\tracingpages=1` e `\tracingpages=0`.

In questo capitolo si sono dati i comandi per il tracciamento delle pagine durante il corpo di pagine precedenti (in formato B5). I messaggi ottenuti nel file `.log` sono ridotti all'osso:

```
[610]
% t=470.0 plus 9.0 minus 8.0 g=466.0 b=12 p=0 c=12#
% t=482.0 plus 10.0 minus 8.0 g=466.0 b=* p=3000 c=*
 [611]
%% goal height=466.0, max depth=5.0
% t=10.0 g=466.0 b=10000 p=3000 c=100000#
% t=22.0 g=466.0 b=10000 p=0 c=100000#
% t=34.0 g=466.0 b=10000 p=0 c=100000#
% t=46.0 g=466.0 b=10000 p=0 c=100000#
% t=58.0 g=466.0 b=10000 p=0 c=100000#
% t=70.0 g=466.0 b=10000 p=0 c=100000#
% t=82.0 g=466.0 b=10000 p=0 c=100000#
% t=94.0 g=466.0 b=10000 p=0 c=100000#
% t=106.0 g=466.0 b=10000 p=0 c=100000#
% t=118.0 g=466.0 b=10000 p=0 c=100000#
% t=130.0 g=466.0 b=10000 p=3000 c=100000#
% t=142.0 g=466.0 b=10000 p=0 c=100000#
% t=154.0 plus 1.0 g=466.0 b=10000 p=3000 c=100000#
% t=166.0 plus 1.0 g=466.0 b=10000 p=0 c=100000#
% t=178.0 plus 1.0 g=466.0 b=10000 p=0 c=100000#
% t=190.0 plus 1.0 g=466.0 b=10000 p=3000 c=100000#
% t=202.0 plus 1.0 g=466.0 b=10000 p=0 c=100000#
% t=214.0 plus 2.0 g=466.0 b=10000 p=3000 c=100000#
% t=226.0 plus 2.0 g=466.0 b=10000 p=0 c=100000#
% t=238.0 plus 2.0 g=466.0 b=10000 p=0 c=100000#
% t=250.0 plus 2.0 g=466.0 b=10000 p=0 c=100000#
% t=262.0 plus 2.0 g=466.0 b=10000 p=0 c=100000#
% t=274.0 plus 2.0 g=466.0 b=10000 p=0 c=100000#
% t=286.0 plus 2.0 g=466.0 b=10000 p=0 c=100000#
```

```
% t=298.0 plus 2.0 g=466.0 b=10000 p=0 c=100000#
% t=310.0 plus 2.0 g=466.0 b=10000 p=0 c=100000#
% t=322.0 plus 2.0 g=466.0 b=10000 p=0 c=100000#
% t=334.0 plus 2.0 g=466.0 b=10000 p=0 c=100000#
% t=346.0 plus 2.0 g=466.0 b=10000 p=0 c=100000#
% t=358.0 plus 2.0 g=466.0 b=10000 p=0 c=100000#
% t=370.0 plus 2.0 g=466.0 b=10000 p=0 c=100000#
% t=382.0 plus 2.0 g=466.0 b=10000 p=3000 c=100000#
% t=394.0 plus 2.0 g=466.0 b=10000 p=0 c=100000#
% t=406.0 plus 3.0 g=466.0 b=10000 p=3000 c=100000#
% t=418.0 plus 3.0 g=466.0 b=10000 p=0 c=100000#
% t=430.0 plus 3.0 g=466.0 b=10000 p=0 c=100000#
% t=442.0 plus 3.0 g=466.0 b=10000 p=3000 c=100000#
% t=454.0 plus 3.0 g=466.0 b=6396 p=0 c=6396#
% t=466.0 plus 4.0 g=466.0 b=0 p=3000 c=3000#
% t=478.0 plus 4.0 g=466.0 b=* p=0 c=*
 [612]
```

Possiamo leggere i risultati così:

- Per la pagina indicata qui col numero `[611]` si legge che il totale di altezza al momento di spedire la pagina all'uscita è di 470,0 pt con la componente di allungamento totale nella pagina di 9,0 pt e una componente di accorciamento di 8,0 pt; l'altezza della pagina da raggiungere (il *goal*) è di 466,0 pt; quindi bisogna accorciare l'altezza della pagina di soli 4,0 pt, abbondantemente rientrante negli 8,0 pt di accorciamento disponibile. La bruttezza è perciò molto piccola (vale solamente 12) non ci sono penalità di nessun genere, il costo per l'emissione di quella pagina è quindi pari alla bruttezza e vale 12; un costo minimo che indica che la fine della pagina 611 è avvenuto nel migliore dei modi. Se si andasse ad una riga dopo, la bruttezza diventerebbe "infinita" (`b=*`) e il costo infinito.

- Per la pagina `[612]` si legge il costo riga per riga: alla penultima riga della pagina si sono raggiunti 454,0 pt, con pochissimo allungamento accumulato, di soli 3,0 pt e non si dispone di nessuna componente di accorciamento. L'altezza della pagina da raggiungere è di 466,0 pt (come prima, evidentemente) ma i 12,0 pt mancanti devono essere recuperati dal piccolissimo allungamento disponibile di soli 3,0 pt; dunque la componente di bruttezza è piuttosto pesante, 6396, ma non infinita; per cui, in mancanza di penalità, il costo per emettere la pagina è piuttosto alto: 6396. In effetti quello sarebbe il costo per emettere la pagina una riga prima della riga orfana in fondo alla pagina. Quella riga in effetti è alta esattamente 12,0 pt, quindi il punto scelto dal programma per emettere la pagina non ha nessuna bruttezza, ma ha una penalità positiva pari alla penalità di default per le righe orfane pari a 3000, quindi abbastanza minore di quella corrispondente alla bruttezza raggiunta alla penultima riga. La pagina, quindi è stata

spezzata nel punto di minimo costo (c=3000) mentre una riga dopo si sarebbe ottenuto un costo grandissimo.

Si noti che avendo caricato il pacchetto `trace` *in questo testo il tracciamento delle pagine cessa con una sola informazione raggiunta verso la fine della pagina nel punto in cui si è specificato il comando di tracciamento e cessa da solo senza bisogno di specificare un nuovo comando con il valore nullo.*

Invece per tracciare il processo di composizione della formazione del capoverso precedente si ha il tracciato seguente, decisamente più complicato da decifrare:

```
@firstpass
[]\T1/lmr/m/it/10
Si noti che avendo caricato il pacchetto \T1/lmvtt/m/sl/10
trace[] \T1/lmr/m/it/10 in questo testo il tracciamento
@ via @@0 b=0 p=0 d=100
@@1: line 1.2 t=100 -> @@0
delle pagine cessa con una sola informazione raggiunta
 verso la fine della pagina
@ via @@1 b=0 p=0 d=100
@@2: line 2.2 t=200 -> @@1
nel punto in cui si è specificato il comando di tracciamento
e cessa da solo
@ via @@2 b=23 p=0 d=1089
@@3: line 3.1 t=1289 -> @@2
senza
@ via @@2 b=0 p=0 d=100
@@4: line 3.2 t=300 -> @@2
bisogno di specificare un nuovo comando con il valore nullo.
@\par via @@3 b=0 p=-10000 d=100
@\par via @@4 b=0 p=-10000 d=100
@@5: line 4.2- t=400 -> @@4
```

Il tracciato comincia con `@firstpass` che indica che le informazioni mostrate indicano la ricerca dei punti migliori di divisione in righe al primo passaggio, senza cioè usare punti di cesura, ma cercando i fine riga solo fra gli spazi interparola. Vi sono indicati i cambiamenti di font nelle righe via via prese in considerazione.

Il primo fine riga con un demerito totale paria a 100 viene trovato dopo la parola “tracciamento”; analogamente il secondo fine riga viene trovato dopo la parola “pagina” di nuovo con un demerito pari a 100; nella terza riga viene provata l’interruzione dopo la parola “solo”, ma i demeriti ammontano a 1089; invece andando avanti e prendendo in considerazione anche la parola “senza” i demeriti ammontano solo al valore 100; questo è quindi accettato come terzo fine riga possibile. Il quarto fine riga avviene alla fine del capoverso dove la penalità vale -10 000, e quindi non ci sono problemi nel terminare la riga alla fine del capoverso (ovviamente!). Nelle ultime due righe vengono riassunti i vari demeriti totali raggiunti attraverso i vari possibili punti di fine riga, e viene

scelto il percorso (segnato dai vari segnaposto `@@1, @@2, ecc.`) che porta al minimo della somma totale dei demeriti pari a 400. Ovviamente in capoversi più complessi, dove sia necessaria anche la seconda passata per cercare i possibili punti di cesura usando la sillabazione, il tracciato potrebbe essere molto più complesso, ma con un po' di pazienza sarebbe decifrabile come si è fatto per questo capoverso.

In realtà la frase che forma il capoverso che abbiamo tracciato è solo una mezza verità; per il tracciamento delle pagine è vera; per il tracciamento dei capoversi è falsa. Per avere il tracciamento completo come abbiamo mostrato sopra è necessario evitare di caricare il pacchetto *trace*, perché appunto configura i comandi di tracciamento in modo troppo ridotto per disporre di un buon tracciamento di pagine e di capoversi.

Chi scrive peraltro riconosce che in generale questi due tracciamenti, ricchissimi di informazioni, sono meno indispensabili di quelli che si ottengono con i comandi `\traceon` e `\traceoff`; chi scrive di solito riesce a capire i difetti di formazione dei capoversi e delle pagine senza ricorrere al loro tracciamento, ma leggendo solo la scarne informazioni associate agli avvisi di "Overfull" box verticali o orizzontali. Naturalmente questa è una questione di preferenze personali oltre che di cose che vengono scritte nei vari tipi di file sorgente quando provocano messaggi di errore o avvisi di cattiva composizione.

28.10 Conclusioni

Benché siano fortemente illuminanti, i comandi di tracciamento per le pagine e per i capoversi vengono usati raramente; i comandi `\traceon` e `\traceoff` vengono usati non tanto spesso, ma sono la vera ultima risorsa a cui accedere quando non si riesce a capire dai messaggi d'errore come eliminare gli errori che il programma di composizione si "ostina" a commettere. Inoltre è quanto mai opportuno leggere attentamente i messaggi d'errore; spesso la spiegazione dell'errore è descritta dal messaggio, ma con parole che non ci sono familiari o che non sappiamo comprendere. Solo la pratica insegna a capire questi messaggi, ma se non si comincia mai a usare gli strumenti descritti qui, non si riuscirà mai a capire il messaggi d'errore fino in fondo.

Nel TEXbook al capitolo 27, dedicato alla soluzione delle condizioni di errore, Knuth termina con queste due citazioni[2]:

Who can understand his errors?
— *Psalm 19:12* (c. 1000 B.C.)

It is one thing, to shew a Man that he is in an Error,
and another, to put him in possession of Truth.
— JOHN LOCKE, *An Essay Concerning Humane Understanding* (1690)

[2]Prima citazione: "Chi riesce a capire i propri errori?". Seconda citazione: "Una cosa è mostrare all'uomo che è in errore, e una differente metterlo in condizione di conoscere la verità."

Senza scendere troppo in considerazioni filosofiche, il problema dell'errore non solo era stato messo nero su bianco fin dai tempi in cui si ritiene che sia stata scritta la Bibbia, ma era anche un argomento dibattuto nel periodo pre-illuministico quando si evidenziava la differenza fra riconoscere l'errore rispetto a correggerlo.

Capitolo 29

Riepilogo della sintassi di LaTeX

In questo capitolo si riepilogherà in modo molto succinto la sintassi di tutti (o quasi) i comandi LaTeX, anche quelli non descritti nei capitoli del testo. L'ordine del materiale segue l'appendice C del manuale di Leslie Lamport, il creatore di LaTeX. Non si tratta però di una semplice traduzione, anche se la materia sembra esposta nello stesso ordine; è una esposizione ragionata e commentata, anche alla luce di informazioni rese disponibili dopo la pubblicazione di quel manuale.

Parentesi graffe e quadre Le parentesi graffe generalmente racchiudono o gli argomenti obbligatori dei comandi oppure un insieme di testo e di comandi e istruzioni che formano un gruppo; i comandi e le istruzioni mantengono la loro efficacia solo all'interno del gruppo, tranne alcuni comandi il cui effetto è globale; questi comandi sono:

`\newcounter`	`\pagenumbering`	`\newlength`	`\hyphenation`
`\setcounter`	`\thispagestyle`	`\newsavebox`	
`\addtocounter`	`\pagecolor`	`\newtheorem`	

Le parentesi quadre delimitano gli argomenti facoltativi di molti comandi; bisogna stare attenti alle possibili contraddizioni fra queste parentesi e le parentesi quadre che compaiono nel testo; queste potrebbero essere scambiate per quelle. Al fine di evitare queste incomprensioni è opportuno racchiudere le parentesi quadre testuali (o le frasi racchiuse fra parentesi quadre) fra parentesi graffe. I comandi con cui potrebbero succedere queste incomprensioni sono i seguenti:

`\\`	`\linebreak`	`\nolinebreak`	`\newcounter`	`\twocolumn`
`\item`	`\pagebreak`	`\nopagebreak`	`\newtheorem`	`\suppressfloats`

Si scriverà per esempio

```
\begin{itemize}
\item {[omissis]} il conte disse allora: "<Ah, ah!">
```

```
\item[{[omissis]}] serve per indicare
                   l'omissione di una parte di testo.
...
```

Gli argomenti degli ambienti Quando si apre un ambiente, solo il comando di apertura può ricevere degli argomenti, mentre il comando di chiusura non ne può accettare alcuno. Se l'ambiente ha un nome che contiene un asterisco finale, questo va ripetuto anche nel comando di chiusura; l'asterisco non è un argomento, ma fa parte integrante del nome.

Comandi fragili e robusti Quasi tutti i comandi di LaTeX sono robusti, nel senso che eseguono quel che devono eseguire senza che questa operazione possa essere modificata dalla posizione che essi hanno, per esempio negli argomenti di altri comandi. Sono fragili gli altri, e per evitare che possano combinare pasticci, bisogna proteggerli con `\protect` premesso al loro nome. Il LaTeX 3 Team sta facendo l'impossibile per eliminare e/o modificare i pochi comandi fragili che rimangono, tuttavia ce ne sono ancora alcuni.

I comandi fragili si possono comportare male quando sono contenuti fra gli argomenti mobili di altri comandi; sono argomenti mobili quelli che devono venire elaborati in fasi successive, per esempio quando devono essere scritti nei file ausiliari e poi riletti ed elaborati in una fase successiva quando si rilancia LaTeX.

I comandi con argomenti mobili sono tipicamente quelli che scrivono qualcosa che deve poi comparire negli indici generale e/o analitico; nei glossari, nelle liste delle figure e/o delle tabelle, eccetera; gli argomenti di `\footnote` e di `\footnotetext`; gli argomenti di `\typein` e di `\typeout` che consentono di creare dei file `.tex` interattivi; il contenuto dell'ambiente *letter* definito all'interno della classe *letter*; l'argomento di `\thanks` usato tipicamente nei titoli, specialmente per indicare le afferenze degli autori; l'argomento facoltativo di `\bibitem`; l'argomento delle *@-espressioni* nella definizione degli incolonnamenti degli ambienti *tabular* e *array*.

È opportuno usare `\protect` con parsimonia, per non correre il rischio di inserirlo dove proprio non ci vuole; conviene inserirlo a posteriori quando l'esecuzione della compilazione mostra errori irrecuperabili, spesso di difficile diagnosi, ma altrettanto spesso dovuti alla fragilità di qualche comando.

**Il comando ** Il comando `\\` serve sia per andare a capo durante la composizione del testo normale, eventualmente contenuto dentro ambienti che specificano il tipo di composizione (testi in display), sia nella composizione di tabelle o matrici, quando si vuole terminare una riga di quella struttura. I due usi possono entrare in collisione, specialmente nelle colonne definite con i descrittori di colonna `p`, `m` o `b`. Per evitare che un comando `\\` usato con il testo che deve venire composto in una colonna `p`, `m` o `b` di una tabella possa venire male interpretato come comando relativo alla tabella, basta racchiudere l'intero contenuto

della cella dentro un gruppo formato dalla coppia di parentesi graffe `{...}`; alternativamente si può usare il comando `\newline`.

Quando LaTeX compone un capoverso si eviti di usare due comandi `\\` di seguito; si usi invece l'argomento facoltativo per aggiungere altro spazio; la sintassi del comando, lo si ricorda, è:

```
\\[⟨spazio⟩]
\\*[⟨spazio⟩]
```

dove la versione asteriscata impedisce un salto pagina prima della nuova riga; a causa di queste diverse modalità di andare a capo, il comando `\\` è fragile.

Gli ambienti (anche nelle versioni asteriscate) e i comandi che accettano il comando `\\`, a parte il testo normale, sono i seguenti:

`\shortstack`	`\author`	`\title`	`\date`
eqnarray	*tabbing*	***array***	***tabular***

Gli ambienti *array* e *tabular* sono stati messi in evidenza perché, insieme con gli altri ambienti per comporre materiale tabulare, bisogna evitare che il segno di 'a capo' `\\` del testo possa essere frainteso per un comando di fine riga `\\` della tabella; questa possibile confusione è particolarmente insidiosa se la parte testuale viene composta con una delle dichiarazioni `\centering`, `\raggedright` o `\raggedleft`. Per evitare di confondere il significato di `\\`, si suggerisce di usare per l''a capo' testuale il comando `\newline`.

Inoltre se si specifica uno ⟨*spazio*⟩ come argomento di `\\` all'interno di una tabella o di una matrice, e se questo è troppo piccolo, potrebbe non manifestarsi nessun effetto visibile, perché il piccolo spazio viene assorbito dalla profondità dello `\strut` che compare in ogni riga delle tabelle e delle matrici.

29.1 La struttura del documento

```
\begin{filecontents}{⟨nome file⟩}
⟨contenuto del file⟩
\end{filecontents}
...
\documentclass[⟨lista di opzioni⟩]{⟨classe⟩}
⟨preambolo⟩
\begin{document}
⟨testo del documento⟩
\end{document}
```

La dichiarazione `\documentclass` può essere preceduta da uno o più ambienti *filecontents* che servono, in particolar modo, quando si inviano i file sorgente ad altri collaboratori, per specificare i pacchetti non standard da cui dipende la composizione del documento. Questi file vengono estratti dal file sorgente la prima

volta che il documento viene elaborato nella nuova situazione e memorizzati con il loro ⟨*nome file*⟩ nella cartella corrente, cosicché quando la compilazione comincia per davvero, LaTeX dispone di tutti i file di cui ha bisogno, compresi, appunto, i file non standard allegati al file sorgente.

29.2 Periodi e capoversi

29.2.1 Periodi

Apostrofi e virgolette

apostrofo: `'`
virgolette semplici rialzate: `‘`⟨*testo*⟩`'`
virgolette doppie rialzate: `‘‘`⟨*testo*⟩`''`
virgolette caporali: `"<`⟨*testo*⟩`">` oppure con encoding `T1`: `<<`⟨*testo*⟩`>>`

Lineati

lineetta: `-`
lineato medio per intervalli numerici: `--`
lineato lungo per gli incisi e per i dialoghi: `---`

Spazi

spazio fine: `\,`
spazio interparola: `\␣`
spazio interparola che impedisce di andare a capo: `~`
spazio di fine periodo: `\@`

Il comando `\@` inserito dopo una lettera maiuscola e prima di un punto fermo serve per evitare che questo sia scambiato per un punto di abbreviazione. L'interprete, infatti, ritiene che un punto dopo una lettera minuscola sia un punto di fine periodo, mentre un punto dopo una lettera maiuscola indichi una abbreviazione. Si osservi la spaziatura quando si scrive M. Rossi e quando, invece, si scrive m. rossi. In questo secondo caso, lo spazio che segue m. dovrebbe essere due volte più grande dello spazio che segue M. e comunque circa il doppio dello spazio interparola. Si confrontino ancora le due frasi:

Vado all'ufficio postale a pagare l'ICI. Poi torno a casa.
Vado all'ufficio postale a pagare l'ICI. Poi torno a casa.

Nel secondo caso *non* è stato inserito il comando `\@` dopo la parola ICI. La differenza è minima, ma un occhio esercitato la rileva. È per questo che nella composizione della bibliografia, dove le abbreviazioni sono numerose, si usa il prossimo comando `\frenchspacing` che rende uguali tutti gli spazi

dopo i segni di interpunzione, in modo che non ci sia da preoccuparsi se un punto indichi una abbreviazione o un punto fermo.

`\frenchspacing` serve per rendere uguali gli spazi dopo qualsiasi segno di interpunzione.

`\nonfrenchspacing` serve per aumentare gli spazi dopo i punti di interpunzione che marcano la fine della frase o del periodo, in particolare dopo i punti fermi. Questa è la situazione di default; secondo alcuni stimati book designer, la spaziatura alla francese sarebbe da evitare sempre; secondo altri dovrebbe essere l'unica ad essere usata. Si noti che questo intero documento è stato composto con la dichiarazione `\frenchspacing`.

Segni speciali

$	`\$`	%	`\%`	{	`\{`	_	`\_`
&	`\&`	#	`\#`	}	`\}`		

Loghi

`\LaTeX`	Logo di LaTeX
`\LaTeXe`	Logo di LaTeX 2$_\varepsilon$
`\TeX`	Logo di TeX

`\today` produce la data del giorno

`\emph{`⟨*testo*⟩`}` Evidenzia ⟨*testo*⟩ scrivendolo in corsivo all'interno di un contesto in tondo e in tondo all'interno di un contesto corsivo.

`\mbox{`⟨*testo*⟩`}` scrive ⟨*testo*⟩ all'interno di una scatola orizzontale in modo che non possa, per esempio, essere divisa in sillabe in fin di riga; questo comando può essere usato anche in un contesto matematico, ma di fatto, se non ci sono motivi speciali, in matematica si possono usare sia i comandi di scelta dei font da usare in un contesto testuale, per esempio `\textrm`, rinunciando però alla prerogativa delle matematica di scegliere il font della giusta dimensione, oppure i comandi `\text` oppure `\intertext` messi a disposizione dal pacchetto *amsmath*.

29.2.2 Capoversi

Un capoverso si può iniziare o terminare con uno dei seguenti comandi; in ogni caso una o più righe assolutamente bianche, senza neanche i segni di commento, servono per terminare un capoverso.

`\noindent` comincia un capoverso senza inserirne il tipico rientro.

`\indent` comincia un capoverso con il tipico rientro, anche quando normalmente esso non sarebbe inserito.

\par termina un capoverso e non è più necessario lasciare altre righe bianche.

I capoversi vengono composti secondo le seguenti giustezze:

\textwidth è la giustezza normale fissata dal file di classe; se la si vuole cambiare bisognerebbe farlo solo nel preambolo.

\linewidth è la giustezza corrente; può essere inferiore o maggiore del valore generale **\textwidth** a seconda dell'ambiente all'interno del quale si sta componendo.

\columnwidth è la giustezza di una colonna, quando si sta componendo su più colonne.

\columnsep è lo spazio di separazione fra due colonne adiacenti.

\columnseprule è lo spessore del filetto verticale che separa le colonne quando si compone su due o più colonne.

\parindent è l'ammontare del rientro del capoverso.

\baselineskip è lo scartamento stabilito con la scelta del corpo del font che si sta usando, eventualmente modificato dal comando seguente.

\linespread{⟨*fattore*⟩**}** modifica lo scartamento mediante il coefficiente moltiplicativo ⟨*fattore*⟩; per attivare questa specifica deve essere impartito il comando **\selectfont** in modo esplicito o implicito (è implicito se si specifica un altro comando di cambiamento di font).

\lineskip interlinea aggiuntiva da aggiungere fra due righe adiacenti per evitare che si sovrappongano.

\lineskiplimit valore minimo della distanza fra due righe successive, scendendo sotto il quale TeX introduce interlinea supplementare pari a **\lineskip**.

\parskip rappresenta uno spazio di interlinea (contrografismo) supplementare da inserire prima di ogni nuovo capoverso; generalmente questo spazio è nullo; talvolta è rappresentato da una *lunghezza elastica* (vedi più avanti nel paragrafo 29.8.2) dotata di un valore naturale nullo, ma di un piccolo allungamento, talvolta anche di un piccolo accorciamento, che consente di giustificare le pagine affacciate anche se contengono un numero di righe leggermente diverso. Questo normalmente succede quando, per esempio, una formula di una certa dimensione verticale o un nuovo paragrafo deve cominciare verso la fine di una pagina, ma se non c'è abbastanza posto, l'oggetto viene spostato alla pagina successiva. La pagina rimarrebbe 'mozza' e, per evitare questo inestetismo, il programma di composizione sfrutta l'elasticità di questi contrografismi per distribuire le righe della pagina in modo che le due pagine affacciate terminino allo stesso livello.

`\enlargethispage{`⟨*lunghezza*⟩`}` allunga la pagina della ⟨*lunghezza*⟩ specificata; per ovvi motivi conviene che questa lunghezza si specificata con un numero intero e piccolo (1 o 2) di righe di testo che, in fondo alla pagina, richiedono un numero intero di righe. Il comando asteriscato `\enlargethispage*{`⟨*lunghezza*⟩`}`, prima di allungare la pagina, cerca di comprimere i contrografismi verticali presenti nella pagina stessa al fine di far stare più materiale nella pagina; in generale è più conveniente usare il comando asteriscato.

29.2.3 Note in calce

`\footnote[`⟨*numero*⟩`]{`⟨*testo*⟩`}`

Se viene specificato il ⟨*numero*⟩ non vi si può fare riferimento in modo simbolico attraverso i comandi `\ref`, `\label`, eccetera. Il contatore `footnote` viene rappresentato con un numero a esponente, tranne che nelle note interne all'ambiente *minipage* dove le note sono numerate con lettere minuscole, sempre a esponente. In certe circostanze è possibile separare il comando che inserisce il richiamo della nota dal comando che effettivamente la scrive:

`\footnotemark[`⟨*numero*⟩`]`
...
`\footnotetext[`⟨*numero*⟩`]{`⟨*testo*⟩`}`

Per i parametri stilistici di composizione si veda il prossimo paragrafo.

29.2.4 Note marginali

`\marginpar[`⟨*nota a sinistra*⟩`]{`⟨*nota a destra*⟩`}`

La ⟨*nota a sinistra*⟩ serve facoltativamente per specificare un testo con una modalità di composizione diverso da quello della ⟨*nota a destra*⟩.

I parametri stilistici con cui vengono composte le note in calce e quelle marginali sono le seguenti.

`\footnotesep` è una lunghezza che rappresenta l'altezza dello `\strut` messo all'inizio di ogni nota; aumentando questa altezza ogni nota viene staccata di più dalla precedente.

`\footnoterule` è il comando con il quale vengono collocati nella lista verticale di oggetti che compongono la pagina il filetto orizzontale e gli spazi adiacenti, in modo da marcare la separazione fra il testo e le note in calce; Complessivamente il filetto e gli spazi adiacenti devono avere una altezza totale nulla, quindi bisogna usare anche dello spazio 'negativo'; infatti la definizione di default di questo comando è la seguente:

```
\def\footnoterule{\kern-3\p@ \hrule \@width 2in \kern 2.6\p@}
```

Benché sia scritto con comandi protetti di basso livello, si capisce che prima si arretra di 3 pt, poi si inserisce un filetto lungo due pollici e spesso 0,4 pt (questo è il valore di defualt), e infine si avanza di 2,6 pt: si constata che $-3\,\text{pt} + 0{,}4\,\text{pt} + 2{,}6\,\text{pt} = 0\,\text{pt}$. Si può ridefinire questo comando con `\renewcommand` ma la definizione deve sempre corrispondere ad una serie di oggetti la cui altezza complessiva sia nulla.

`\marginparsep` è la larghezza dello spazio che separa le note marginali dal testo a cui sono affiancate.

`\marginparpush` è la distanza minima fra due note marginali consecutive.

`\marginparwidth` è la giustezza con cui sono composte le note marginali; va da sé che il margine che le deve contenere deve essere sufficientemente largo e deve tenere conto anche di `\marginparsep` e di un buono spazio verso il taglio della pagina affinché dopo il taglio non si debba scoprire che anche parte della nota è stata tagliata.

`\reversemarginpar` è una istruzione booleana che imposta lo scambio del margine nel quale vengono composte le note marginali; agisce solo quando si compone ad una sola colonna, perché a due colonne le note marginali della colonna di sinistra vengono composte comunque a sinistra, e quelle relative alla colonna di destra, a destra di questa.

`\normalmarginpar` ristabilisce la composizione delle note marginali nel margine esterno.

29.2.5 Accenti e simboli speciali

Premesso che con l'uso del pacchetto *inputenc* e della tastiera nazionale la necessità di ricorrere ai comandi per gli accenti è fortemente diminuita, tuttavia è utile saper gestire anche i comandi più insoliti al fine di comporre correttamente anche parole in lingue straniere che fanno uso di segni che non compaiono sulla tastiera nazionale.

I comandi per gli accenti testuali sono raccolti nella tabella 29.1, mentre quelli per il contesto matematico sono raccolti nella tabella 13.7.

29.3 Suddivisione del testo e indici

29.3.1 Comandi di sezionamento

Tutti i comandi di sezionamento propriamente detti, e gli altri comandi simili hanno una sintassi comune:

comando[⟨*titolo breve*⟩]{⟨*titolo lungo*⟩}

Essi sono:

\‘{a}	à	\’{a}	á	\={a}	ā
\v{a}	ǎ	\^{a}	â	\"{a}	ä
\~{a}	ã	\u{a}	ă	\.{a}	ȧ
\H{a}	a̋	\t{oo}	o͡o	\c{t}	ţ
\d{a}	ạ	\b{a}	a̲	\r{u}	ů
\i	ı	\j	ȷ	\oe	œ
\OE	Œ	\ae	æ	\AE	Æ
\aa	å	\AA	Å	\o	ø
\O	Ø	\l	ł	\L	Ł
\ss	ß	?‘	¿	!‘	¡
\dag	†	\ddag	‡	\S	§
\P	¶	\copyright	©	\pounds	£

Tabella 29.1: I comandi per gli accenti testuali e i simboli speciali di molte lingue straniere

```
\part            \chapter     \section        \subsection
\subsubsection   \paragraph   \subparagraph   \caption
```

Il comando `\chapter` non è definito per la classe *`article`*; nessun comando di sezionamento è definito per la classe *`letter`*. Il livello di sezionamento `\part` non è obbligatorio; tutti gli altri comandi devono essere contenuti sotto un comando del livello precedente.

Le parole che eventualmente vengono scritte nelle intestazioni, come ‘Part’ o ‘Parte’, ‘Chapter’ o ‘Chapître’, vengono impostate tramite l’opportuna chiamata al pacchetto *`babel`* con le opzioni per la lingua del documento. In ogni caso esse sono contenute nei comandi `\partname`, `\chaptername`, `\figurename`, `\tablename`, per cui possono venire ridefinite a piacere.

Ogni comando di sezionamento oltre a comporre l’intestazione della divisione specifica mediante il ⟨*titolo lungo*⟩, scrive negli opportuni file accessori il ⟨*titolo breve*⟩ da inserire nei corrispondenti indici e, con certe classi, nelle testatine.

Ogni divisione del documento può venire numerata e può comparire nell’indice; questi due fatti sono controllati dai valori numerici contenuti nei due contatori `secnumdepth` e `tocdepth`; Le sezioni sono numerate se il loro livello è associato ad un numero minore o uguale a quello contenuto in `secnumdepth`; i loro titoli vanno a finire nell’indice se il loro livello corrisponde ad un numero minore o uguale a quello contenuto in `tocdepth`; si veda la tabella 29.2.

Per produrre gli indici si usano i comandi:

```
\tableofcontents    \listoffigures    \listoftables
```

Le informazioni per produrre questi indici sono contenute in tre file ausiliari le cui estensioni sono rispettivamente `.toc`, `.lof` e `.lot`.

Sezionamento	Livello	Macro che contiene il nome
`\part`	−1	`\partname`
`\chapter`	0	`\chaptername`
`\section`	1	
`\subsection`	2	
`\subsubsection`	3	
`\paragraph`	4	
`\subparagraph`	5	
`\caption`	—	`\figurename` `\tablename`

Tabella 29.2: Numeri associati al livello di sezionamento

Le versioni asteriscate dei comandi di sezionamento producono solo l'intestazione non numerata e non producono niente nell'indice né nelle testatine. Il comando `\caption` non possiede la variante asteriscata.

Con alcune classi standard, come *`book`* si possono considerare comandi di sezionamento anche `\frontmatter`, `\maimatter`, `\backmatter`; non c'è dubbio che in un documento complesso come un libro queste tre parti costituenti il documento debbano essere distinte e i tre comandi suddetti eseguono questo sezionamento; apparentemente non scrivono nulla nel documento, ma queste tre dichiarazioni modificano in modo sostanziale il modo di comporre il documento stesso, in particolare il comportamento dei comandi di sezionamento.

1. Il comando `\frontmatter` imposta la numerazione delle pagine con numeri romani minuscoli; fa si che i comandi di sezionamento *non asteriscati* non producano titoli numerati, ma i loro titoli vengono inseriti nell'indice.

2. Il comando `\mainmatter` reimposta lo stile di numerazione delle pagine in cifre arabe e consente che i comandi di sezionamento *non asteriscati* abbiano il loro titoli numerati ed eventualmente inseriti nelle testatine se lo stile delle pagine lo prevede e se i contatori `tocdepth` e `secnumdepth` lo consentono.

3. Il commando `\backmatter` non reimposta la numerazione delle pagine, quindi la numerazione in cifre arabe prosegue quella della parte principale del testo, ma di nuovo i comandi di sezionamento *non asteriscati* non sono più numerati ma possono avere i loro titoli nelle testatine, come nella parte iniziale. A questo proposito ci sono alcuni che ritengono che le appendici facciano parte della parte finale (quindi vi appaiano solo con i loro titoli senza la consueta numerazione letterale), e altri ritengano che, quando le appendici sono più di una, esse debbano essere numerate (con la consueta numerazione letterale) e debbano apparire nell'indice, e perciò debbano far parte della parte principale.

La descrizione delle caratteristiche di questi speciali comandi di sezionamento implica che se si usano i comandi di sezionamento *asteriscati*, `\chapter*`, `\section*`, eccetera, si comportano in modo diverso nel senso che producono sezioni non numerate i cui titoli non vanno a finire nelle testatine. Quando si usano classi che non prevedono l'uso di `\frontmatter` e compagni, ci sono due modi per aggirare questi due "scogli":

1. Si possono impostare i contatori `secnumdepth` e `tocdepth` in modo da replicare il funzionamento di `\frontmatter`:

   ```
   \setcounter{secnumdepth}{-2}
   \setcounter{tocepth}{2}
   ```

   In questo modo *non* bisogna usare i comandi asteriscati, ma bisogna usare i comandi normali *non asteriscati*.

2. Si possono usare i comandi asteriscati, ma bisogna fornire con ulteriori comandi le informazioni che servono; per esempio con `\chapter*` si può fare così:

   ```
   \chapter*{⟨Titolo del capitolo non numerato⟩}
   \addcontentsline{toc}{chapter}{⟨Titolo del capitolo non numerato⟩}
   \markboth{⟨Titolo del capitolo non numerato⟩}{⟨Titolo del capitolo non numerato⟩}
   ```

   Per i dettagli si veda quanto scritto qui di seguito a proposito del comando `\addconttentsline`; questo comando serve per scrivere il suo contenuto nell'indice con lo stile delle righe riferite ai capitoli; se non interessa inserire la voce nell'indice, non si usa l'intera riga.

   Invece il fatto che `\markboth` contenga lo stesso titolo sia per le testatine di sinistra sia per quelle di destra, nasce dalla considerazione che un capitolo non numerato difficilmente verrebbe diviso in sezioni; se lo fosse, anche queste dovrebbero essere ottenute con comandi asteriscati e dovrebbero venire trattate in modo simile a quello dei capitoli:

   ```
   \section*{⟨Titolo del paragrafo non numerato⟩}
   \addcontentsline{toc}{section}{⟨Titolo del paragrafo non numerato⟩}
   \markright{⟨Titolo del paragrafo non numerato⟩}
   ```

   In questo modo eventuali sezionamenti del capitolo non numerato avrebbero nelle testatine di destra il titolo del paragrafo non numerato, ma resterebbero inalterate le testatine di sinistra.

Il comando `\addcontentsline` ha la seguente sintassi:

```
\addcontentsline{⟨estensione⟩}{⟨livello⟩}{⟨voce⟩}
```

dove ⟨*estensione*⟩ è l'estensione del file ausiliario per l'indice specifico; ⟨*livello*⟩ è il nome (non il numero) del livello di sezionamento, espresso dallo stesso nome del comando, senza però il backslash; la ⟨*voce*⟩ è quanto viene scritto nel file indice. Generalmente quanto viene scritto nell'indice è costituito dall'espressione

```
\protect\numberline{⟨numero⟩}{⟨titolo⟩}
```

dove ⟨*numero*⟩ è il numero della sezione, per esempio 3.8 per l'ottavo paragrafo del capitolo tre; va da sé che una sezione non numerata ha questo campo vuoto. Il ⟨*titolo*⟩ non è altro che il titolo (breve) della sezione.

Per le sezioni non numerate lasciare vuoto il campo riservato al ⟨*numero*⟩ implica che il ⟨*titolo*⟩ viene incolonnato con gli altri titoli delle sezioni numerate; se invece si omette tutta la parte `\protect\numberline{`⟨*numero*⟩`}`, allora il ⟨*titolo*⟩ viene allineato al margine sinistro. Si veda, per esempio, nell'indice generale di questo testo dalla voce 'Bibliografia' all'ultimo indice analitico.

Per l'indice analitico e il glossario, i vari elementi vengono raccolti mediante i rispettivi comandi `\index` e `\glossary`, ma il contenuto del loro argomento varia molto a seconda di come si intende usare l'informazione raccolta. In ogni caso nessuna informazione viene raccolta se nel preambolo mancano i comandi `\makeindex` e, rispettivamente, `\makeglossary`; l'indice analitico e/o il glossario non vengono composti se non si dà esplicitamente il comando di comporli, di solito attraverso comandi messi a disposizione da pacchetti esterni. Altrimenti bisogna usare gli ambienti *index* e *glossary* specificando a mano il nome dei file che contengono l'indice o il glossario elaborati adeguatamente.

29.4 Classi, pacchetti e stili delle pagine

29.4.1 Classe del documento

La scelta della classe va eseguita con il comando:

```
\documentclass[⟨opzioni⟩]{⟨classe⟩}
```

Le classi standard sono *book*, *report*, *article*, *letter*, *slides*, *minimal*, *proc*, *ltxdoc*; queste sono affiancate dalle numerosissime altre classi non standard che popolano gli archivi internazionali.

Le opzioni standard sono le seguenti:

10pt, definisce il corpo di default per il documento; però può venire trasmessa anche ad altri pacchetti, fra i quali *type1ec* il quale con questa opzione ricava i disegni di tutti i caratteri dai font in corpo 10 pt, ingrandendoli o rimpicciolendoli a seconda delle richieste del compositore. Questo modo di procedere è accettabile solo quando si sta lavorando con font vettoriali. Il file di uscita risulta meno ingombrante, la qualità peggiora un poco, ma questa riduzione è visibile ad occhio nudo. Tra l'altro si noti la differenza di resa dei font ingranditi o rimpiccioliti; qui si mostra lo stesso testo

riportato al corpo 10 partendo da un font di corpo 5 e da un font di corpo 17: **partendo da un font di corpo 5** e partendo da un font di corpo 17. L'effetto opposto si otterrebbe partendo da un font di corpo 10 riducendolo al corpo 5 o ingrandendolo al corpo 17.

11pt, *12pt*, servono per scegliere il corpo da 11 pt o da 12 pt come corpo normale.

letterpaper, *legalpaper*, *executivepaper*, *a4paper*, *a5paper*, *b5paper*, servono per scegliere il supporto della carta secondo i formati specificati con le sigle suddette, precisamente:

letter	8,5 in × 11 in	A4	210 mm × 297 mm
legal	8,5 in × 14 in	A5	148 mm × 210 mm
executive	7,25 in × 10,5 in	B5	176 mm × 250 mm

Siccome l'opzione di default è *letterpaper* per scrivere in Europa è sempre necessario specificare un formato ISO, quasi sempre *a4paper*.

Le dimensioni così specificate vengono passate anche al file `.pdf` nel caso che si usi pdfLaTeX, in quanto quel formato ha bisogno di conoscere le dimensioni del supporto di uscita. Il file `.dvi` a stretto rigore non ne ha bisogno, ma poi diventa necessario se questo tipo di file viene convertito nei file `.ps` oppure `.pdf`.

landscape, serve per scambiare fra di loro la larghezza e l'altezza del supporto.

final, *draft*, servono per specificare se si sta componendo la versione finale, oppure una bozza; quando si compone una bozza, vengono evidenziate le righe fuori giustezza con un vistoso rettangolo nero e non vengono importate le figure, ma al loro posto viene segnato solo un rettangolo delle giuste proporzioni con il nome del file grafico scritto dentro.

oneside, *twoside*, servono per specificare se si vuole comporre solo sul recto o anche sul verso della carta.

openright, *openany*, servono per specificare se i capitoli possono iniziare solo sulle pagine di destra oppure su qualunque pagina.

onecolumn, *twocolumn*, servono per specificare si si vuol comporre su un'unica colonna o su due colonne.

notitlepage, *titlepage*, servono per specificare se il titolo del documento va in testa alla prima pagina, nella quale comincia subito anche il testo, oppure se si deve riservare una pagina solo per il titolo.

openbib, serve per comporre la bibliografia (nelle classi che la riconoscono) con uno stile più aperto; generalmente le varie parti del riferimento bibliografico sono trattate come dei brevi capoversi senza rientro; però le loro righe dopo la prima risultano composte con un rientro, rispetto al margine sinistro, specificato mediante il parametro `\bibindent`.

leqno, serve per specificare che si desidera numerare le equazioni con il numero collocato a filo del margine sinistro della gabbia del testo. Se anche le equazioni vengono allineate a sinistra, potrebbe non esserci abbastanza spazio.

fleqn, serve per specificare che non si desiderano le equazioni centrate, ma si desidera che siano rientrate di una quantità fissa dal margine sinistro; questa quantità viene specificata mediante `\mathindent` che di default imposta al valore della rientranza del margine sinistro delle liste di primo livello. Forse è un po' poco, specialmente se i numeri identificativi delle equazioni (che possono diventare relativamente lunghi quando esse sono numerose) con l'opzione precedente è messo accostato al margine sinistro dello specchio di stampa; si può sempre modificare `\mathindent` con il comando `\setlength`. Va notato, però, che con il valore di default di `\mathindent` non vengono introdotti altri valori di rientranza dei margini, e questo tipograficamente va bene, ma con numerose equazioni potrebbe verificarsi l'interferenza dell'etichetta dell'equazione con l'equazione stessa. Il pacchetto *amsmath* evita sempre questa interferenza, perciò direi che sia meglio usare *amsmath* piuttosto di modificare `\masthindent`.

29.4.2 Pacchetti

La scelta del pacchetto si esegue con il comando `\usepackage` con la sintassi

`\usepackage[`⟨*opzioni*⟩`]{`⟨*pacchetto*⟩`}[`⟨*data*⟩`]`

dove ⟨*data*⟩ è una data in forma ISO (*aaaa/mm/gg*) che rappresenta il limite inferiore della data corrispondente alla versione che si vuole usare; nei circa 20 anni da quando esiste LaTeX 2_ε i vari pacchetti sono stati aggiornati così che versioni successive non solo hanno (presumibilmente) meno errori o 'features' degli stessi pacchetti nelle loro versioni precedenti, ma anche funzionalità maggiori. Ecco che specificando la data si può essere sicuri di usare una versione abbastanza recente che presenta le funzionalità desiderate. I pacchetti standard sono:

alltt serve per definire l'ambiente *alltt* dove quasi tutti i caratteri speciali perdono il loro significato specifico e vengono usati come caratteri 'normali'.

amsmath costituisce la principale estensione per gli ambienti matematici; vedi più avanti l'uso dei pacchetti *amssymb* e *amsfonts*.

babel consente di comporre in diverse lingue; attenzione: *babel* sceglie le specifiche della lingua, compresa la sillabazione, ma non è *babel* che carica la sillabazione; vedi il capitolo 25.

color serve per gestire i colori dei font e degli sfondi.

graphics serve per gestire le manipolazioni grafiche. Benché sia il pacchetto che svolge il lavoro richiesto, per l'utente è più comodo riferirsi al pacchetto seguente.

graphicx gestisce le operazioni grafiche e gli effetti speciali con una interfaccia utente più efficace; in realtà è *graphicx* che si preoccupa di caricare *graphics* e di passargli i comandi tradotti nella sua sintassi specifica.

ifthen permette di eseguire un fine controllo logico con una interfaccia utente più semplice di quella che offrono i comandi primitivi di TEX. Si veda più avanti la spiegazione del perché sia meglio usare altri pacchetti.

latexsym consente di usare i simboli speciali di LATEX 2.09; questo pacchetto è più per compatibilità con il passato che altro; tutti quei simboli sono accessibili anche con il pacchetto *amssymb* che consente l'accesso ai font della American Mathematical Society descritti nel pacchetto *amsfont*.

makeidx serve per gestire comodamente la creazione dell'indice analitico. Si possono usare in alternativa i pacchetti *imakeidx* o *indextools* che gestiscono anche la produzione di indici analitici multipli e consentono di sfruttare la proprietà dei sistemi moderni di lanciare comandi di sistema cosicché gli indici analitici possono essere prodotti direttamente con una sola esecuzione di pdflatex o degli altri programmi di composiione.

pict2e estende le capacità grafiche dell'ambiente *picture* di LATEX. Queste estensioni sono disponibili solo dal 2003, e il pacchetto ha raggiunto una situazione abbastanza stabile solo dal 2004; nel 2009 e poi nel 2011 gli sono state ulteriormente estese le funzionalità; si vede dunque come l'opzione della ⟨*data*⟩ non sia una cosa tanto insolita da usare.

showidx consente di scrivere le voci raccolte con `\index` direttamente sulla pagina dove sono state trovate; durante la lavorazione di un documento questa opzione si rivela utile per selezionare e confrontare le varie voci, sia per ridurre i duplicati sia per scegliere i riferimenti più adatti.

29.4.3 Stili delle pagine

Ogni classe inizializza lo stile delle pagine normali secondo le prescrizioni stilistiche generali della classe stessa. Il comando per scegliere lo stile è

`\pagestyle{`⟨*stile*⟩`}`

Il comando

`\thispagestyle{`⟨*stile*⟩`}`

imposta lo stile della pagina solo per la pagina corrente.

Gli stili predefiniti nelle classi standard sono i seguenti.

`plain` contiene solo il piedino con il numero della pagina.

`empty` sia il piedino sia la testatina sono vuoti.

`headings` il piedino è vuoto ma la testatina è elaborata e automaticamente contiene sul verso delle pagine scritte con l'opzione *twoside* il titolo (breve) del capitolo e nella testatina del recto il titolo (breve) del paragrafo attivo all'inizio della pagina.

Questi titoli sono forniti dai comandi di sezionamento in modo automatico; se si desiderano testatine specifiche, che contengano elementi diversi o titolini correnti diversi bisogna usare i comandi `\markright` e `\markboth`, oppure bisogna servirsi di pacchetti di estensione come, per esempio, *`fancyhdr`*. Oppure bisogna servirsi di classi diverse da quelle standard.

`myheadings` in senso generale è simile allo stile `headings`, salvo che le informazioni da inserire nelle testatine debbono essere fornite esplicitamente mediante i comandi `\markright` e `\markboth`.

La sintassi di questi comandi è la seguente:

```
\markright{⟨titolino di destra⟩}
\markboth{⟨titolino di sinistra⟩}{⟨titolino di destra⟩}
```

La numerazione delle pagine viene specificata con il comando

```
\pagenumbering{⟨stile di numerazione⟩}
```

dove gli stili di numerazione disponibili sono i seguenti:

`arabic` serve per la normale numerazione con cifre arabe.

`roman` serve per la numerazione romana in lettere minuscole.

`Roman` serve per la numerazione romana in lettere maiuscole

`alph` serve per la numerazione alfabetica in lettere minuscole.

`Alph` serve per la numerazione alfabetica in lettere maiuscole.

`fnsymbol` serve per rappresentare i numeri con la stessa sequenza di simboli usata per le note, per esempio l'asterisco, la spada, la doppia spada e via di seguito. I simboli in totale sono nove, quindi non è possibile numerare più di 9 cose. Generalmente questo tipo di numerazione non viene usato per le pagine, ma lo si usa abbastanza spesso per le note, in particolare per quelle del frontespizio.

Difficilmente si useranno numerazioni alfabetiche perché il numero totale delle pagine da numerare potrebbe superare 26, ma per le pagine le numerazioni arabe e romane vengono usate spesso.

Lo stile della pagina è vistosamente differente se si compone su una sola colonna o su due o più colonne; le classi standard consentono al massimo due colonne, ma mediante il pacchetto *multicol* non è difficile aumentare il numero delle colonne.

Quando però si usano le classi standard e si specifica di comporre ad una sola colonna, è possibile comporre alcune pagine su due colonne, consentendo di comporre anche una intestazione ad una colonna. Al contrario se si è specificata l'opzione per comporre su due colonne, allora è possibile comporre parte delle pagine su una sola colonna. I comandi da usare sono:

```
\twocolumn[⟨intestazione ad una colonna⟩]
\onecolumn
```

Entrambi i comandi iniziano sempre una nuova pagina poi cominciano a comporre come dice il loro nome.

Ovviamente lo stile della pagina dipende anche dalla giustezza e dalla sua relazione con il formato della carta, quindi dei margini. Le giustezze orizzontali e verticali sono specificate mediante le lunghezze `\textwidth` e `\textheight` e i margini sono specificati mediante le lunghezze `\oddsidemargin` per le pagine di destra, `\evensidemargin` per le pagine di sinistra, `\topmargin` per il margine superiore; i margini laterali sono comunque i margini sinistri, e le parole 'odd' e 'even' (dispari o pari) si riferiscono alla posizione della pagina in relazione alla sua numerazione, sapendo che il verso di tutte le pagine è sempre pari e il recto sempre dispari. Questi margini sono riferiti allo spigolo superiore sinistro del foglio di carta e tutti sono 'un pollice' di meno di quello che ci si aspetterebbe; questo dipende dal fatto che inizialmente era previsto che tutti i driver di uscita avrebbero inserito di default un pollice di margine.

Non è il caso di mettersi a giocare con questi margini; se proprio si deve modificare l'aspetto geometrico di una pagina, è meglio usare una classe che consenta di farlo, oppure il pacchetto *geometry*.

29.4.4 Il frontespizio

È vero che se uno deve comporre un frontespizio in modo decoroso, è opportuno che studi il layout e la scelta del corpo e dello stile dei font in modo professionale. Tuttavia nei rapporti e negli articoli anche la modesta composizione di default operata dalla classi standard di LaTeX può essere accettabile.

Bisogna ovviamente specificare l'opzione *titlepage*; ma bisogna anche specificare nel preambolo un certo numero di informazioni.

`\title{`⟨*titolo*⟩`}` serve per specificare il titolo del documento; se questo titolo è sufficientemente lungo lo si può comporre su più righe inserendo il comando `\\`. Bisogna però ricordarsi di non andare a capo fra un articolo e il nome, fra una preposizione e il suo complemento, fra avverbi brevi, come 'non', e quanto segue; fa parte dell'eleganza della composizione. Riassumendo: nei titoli non si deve mai andare a capo dopo le congiunzioni, gli articoli,

le preposizioni e i brevi avverbi; per evitarlo si deve usare il comando di legatura costituito dalla tilde `~`.

`\author{`⟨*nomi*⟩`}` serve per specificare i nomi degli autori; ma se gli autori sono più di uno bisogna separarli con `\and`; si può andare a capo usando `\\`. Attenzione: per ciascun autore sempre il o i nomi propri prima del o dei cognomi!

`\date{`⟨*testo*⟩`}` il testo di questo comando potrebbe essere una data, come suggerisce il nome del comando, ma potrebbe anche essere il nome di una conferenza, un luogo, qualunque informazione che permetta di identificare il documento oltre al suo titolo e ai suoi autori (i quali potrebbero aver prodotto un articolo con lo stesso titolo ad un altro convegno...).

`\thanks{`⟨*testo*⟩`}` questo comando può essere appeso sia ai nomi degli autori, per esempio per specificarne l'afferenza, sia al titolo, per esempio per associargli il nome dell'ente finanziatore delle ricerca, sia al testo specificato con `\date`. Esso funziona come il comando per inserire delle note a piè di pagina, solo che il riferimento di queste note non è né un numero né una lettera, ma un generico simbolo tratto da una lista che contiene l'asterisco, la spada, la spada doppia, eccetera.

Il tutto viene eseguito dando il comando `\maketitle`. Alternativamente e con risultati esteticamente variabili a seconda del gusto del compositore, si può comporre il frontespizio aprendo l'ambiente *titlepage* e scrivendoci dentro quello che si desidera, dove lo si desidera, con i font che si desiderano; talvolta la cosa produce buoni risultati, talaltra è meglio affidarsi alla composizione di default.

Nei rapporti e negli articoli spesso il frontespizio può contenere anche un breve riassunto; in LaTeX questo viene composto mediante l'ambiente *abstract*. Il riassunto può comparire come prima cosa dell'articolo e se questo è composto su due colonne, il sunto si trova nella prima colonna, subito sotto il titolo e le informazioni composte con `\maketitle`.

Ovviamente classi diverse da quelle standard possono mettere a disposizione del compositore degli ambienti preconfezionati di *titlepage* che dispongono le informazioni desiderate in modo diverso, più elegante, includendo alche il retro del frontespizio, con le informazioni legali, il copyright, gli avvisi di copia riservata o proibita, altre informazioni sulle persone che hanno collaborato a impaginare il testo, a fare i disegni, le fotografie, e simili. Alcune classi consentono anche di inserire nel retro del frontespizio anche il barcode classico a 10 cifre, sia quello moderno a 13 cifre. Bisogna però documentarsi accuratamente in merito alle classi alternative per scegliere quella che fa al caso proprio.

29.5 Testi in display

29.5.1 Citazioni e poemi

Gli ambienti per i testi in display sono:

```
\begin{quote} ⟨testo⟩ \end{quote}
\begin{quotation} ⟨testo⟩ \end{quotation}
\begin{verse} ⟨testo⟩ \end{verse}
```

29.5.2 Liste

Gli ambienti predefiniti per le liste sono:

```
\begin{itemize}
\item[⟨contrassegno⟩] ⟨testo⟩
...
\end{itemize}
```

```
\begin{enumerate}
\item[⟨contrassegno⟩] ⟨testo⟩
...
\end{enumerate}
```

```
\begin{description}
\item[⟨voce da descrivere⟩] ⟨testo di descrizione⟩
...
\end{description}
```

Per i primi due ambienti il ⟨*contrassegno*⟩ è davvero facoltativo; per l'ambiente *description*, benché appaia come argomento facoltativo, in realtà esso è obbligatorio, perché non avrebbe senso dare una descrizione di nulla. Tuttavia se non si specifica nulla, nemmeno le parentesi quadre, si comincia una nuova descrizione senza etichetta che può essere vista come un nuovo capoverso. Non sembra una buona idea sfruttare così male le potenzialità di LaTeX.

L'ambiente principale, con il quale sono costruite tutte le altre liste, però, è l'ambiente *list*, che è completamente configurabile in ogni dettaglio. La sintassi è:

```
\begin{list}{⟨contrassegno di default⟩}{⟨dichiarazioni⟩}
\item[⟨contrassegno personalizzato⟩] ⟨testo⟩
...
\end{list}
```

Il ⟨*contrassegno di default*⟩ è il modo di comporre il contrassegno che *list* produce quando non viene esplicitato un ⟨*contrassegno personalizzato*⟩ con il

comando `\item`. Le ⟨*dichiarazioni*⟩, invece, sono una serie di istruzioni che specificano il modo di comporre eseguito da *list*. Ci sono numerose possibilità che è meglio descrivere una alla volta.

`\topsep` è la distanza che separa il testo precedente alla lista dalla prima voce della lista. La stessa distanza viene posta dopo la chiusura della lista.

`\partopsep` è lo spazio aggiuntivo aggiunto prima e dopo la lista se questa comincia un nuovo capoverso, cioè se nel file sorgente una riga completamente bianca precede il comando di apertura dell'ambiente.

`\itemsep` è lo spazio aggiuntivo che viene messo prima di una voce della lista se questa è preceduta da una riga bianca.

`\parsep` Siccome dentro le liste i capoversi solitamente non vengono rientrati, allora si può usare un piccolo spazio aggiuntivo fra due capoversi appartenenti alla stessa voce; questo spazio di separazione fra i *paragraphs* si chiama appunto `\parsep`.

`\leftmargin` è la rientranza di ogni voce della lista rispetto al testo circostante; liste annidate hanno rientranze sempre maggiori; il file di classe specifica le varie successive rientranze mediante i comandi `\leftmargini`, `\leftmarginii`, `\leftmarginiii` e `\leftmarginiv`.

`\rightmargin` è la distanza orizzontale fra il margine destro della lista corrente e il margine destro del testo che contiene la lista.

`\listparindent` è il rientro extra aggiunto (o anche tolto se ha un valore negativo) alla rientranza di ogni linea di ciascuna voce, tranne la prima riga; tutte le classi standard hanno questo parametro impostato a zero, ma non è impossibile che in altre classi o in liste personalizzate non si possa usare questo parametro.

`\itemindent` è la rientranza della prima riga di ogni voce della lista. Può anche avere un valore negativo.

`\labelsep` è la distanza minima che separa il ⟨*contrassegno*⟩ dal resto del testo nella prima riga di ogni voce.

`\labelwidth` è la larghezza prevista per il ⟨*contrassegno*⟩; a seconda della personalizzazione se questo contrassegno fosse più largo o più stretto di questa larghezza, il contrassegno viene fatto sporgere a sinistra, oppure viene fatto sporgere a destra ma cominciando il testo della voce più a destra in modo da mantenere costante lo spazio fra la fine del ⟨*contrassegno*⟩ e l'inzio della voce.

`\makelabel{`⟨*contrassegno*⟩`}` è la macro che effettivamente compone il contrassegno.

`\usecounter{`⟨*contatore*⟩`}` specifica che il ⟨*contatore*⟩ è quello usato per numerare gli elementi della lista e lo rende usabile con i comandi `\label` e `\ref`.

Vale la pena di provare a leggere la definizione dell'ambiente *enumerate* per capire meglio come si usi l'ambiente *list*. L'esempio che si porta è ripreso dalle definizioni, sparse in diversi file, per l'ambiente *enumerate* della classe *book* quando si compone con il corpo di default di 10 pt. Le definizioni vere e proprie dell'ambiente sono eseguite con comandi di basso livello e/o con comandi di servizio di LaTeX, ma si ritiene che siano abbastanza chiare da non richiedere una spiegazione, mentre se ne daranno per altri aspetti delle definizioni.

```
%                                          Definizioni in latex.ltx
\newcount\@enumdepth \@enumdepth = 0
\@definecounter{enumi}
\@definecounter{enumii}
\@definecounter{enumiii}
\@definecounter{enumiv}
%                                           Definizioni in book.cls
\renewcommand\theenumi{\@arabic\c@enumi}
\renewcommand\theenumii{\@alph\c@enumii}
\renewcommand\theenumiii{\@roman\c@enumiii}
\renewcommand\theenumiv{\@Alph\c@enumiv}
\newcommand\labelenumi{\theenumi.}
\newcommand\labelenumii{(\theenumii)}
\newcommand\labelenumiii{\theenumiii.}
\newcommand\labelenumiv{\theenumiv.}
\renewcommand\p@enumii{\theenumi}
\renewcommand\p@enumiii{\theenumi(\theenumii)}
\renewcommand\p@enumiv{\p@enumiii\theenumiii}
%                                           Definizioni in bk10.clo
\def\@listi{\leftmargin\leftmargini
            \parsep 4\p@ \@plus2\p@ \@minus\p@
            \topsep 8\p@ \@plus2\p@ \@minus4\p@
            \itemsep4\p@ \@plus2\p@ \@minus\p@}
\let\@listI\@listi
\@listi
\def\@listii {\leftmargin\leftmarginii
              \labelwidth\leftmarginii
              \advance\labelwidth-\labelsep
              \topsep    4\p@ \@plus2\p@ \@minus\p@
              \parsep    2\p@ \@plus\p@  \@minus\p@
              \itemsep   \parsep}
\def\@listiii{\leftmargin\leftmarginiii
              \labelwidth\leftmarginiii
              \advance\labelwidth-\labelsep
```

```
              \topsep    2\p@ \@plus\p@\@minus\p@
              \parsep    \z@
              \partopsep \p@ \@plus\z@ \@minus\p@
              \itemsep   \topsep}
\def\@listiv {\leftmargin\leftmarginiv
              \labelwidth\leftmarginiv
              \advance\labelwidth-\labelsep}
\def\@listv  {\leftmargin\leftmarginv
              \labelwidth\leftmarginv
              \advance\labelwidth-\labelsep}
\def\@listvi {\leftmargin\leftmarginvi
              \labelwidth\leftmarginvi
              \advance\labelwidth-\labelsep}
%                                        Definizioni in latex.ltx
\def\enumerate{%
  \ifnum \@enumdepth >\thr@@\@toodeep\else
    \advance\@enumdepth\@ne
    \edef\@enumctr{enum\romannumeral\the\@enumdepth}%
      \expandafter
      \list
        \csname label\@enumctr\endcsname
        {\usecounter\@enumctr\def\makelabel##1{\hss\llap{##1}}}%
  \fi}
\let\endenumerate =\endlist
```

Inizialmente si definiscono il contatore di livello e i quattro contatori per ciascuno dei quattro livelli; si ricorderà infatti che le liste, anche le enumerazioni, possono essere annidate l'una nell'altra, ma solo fino al quarto livello.

Successivamente nella classe *book* si definiscono i modi di rappresentare i vari contatori di enumerazione; il primo contatore viene stampato mediante `\@arabic\c@enumi`, cioè in cifre arabe; nella lista, però, esso viene marcato come il numero seguito da un punto: `\theenumi.` Per il primo livello basta così, ma per i livelli successivi ci vuole un prefisso; quando si cita il terzo elemento della seconda lista annidata, per esempio, non basta la scrittura del valore del secondo contatore, che viene stampato con una lettera mediante `\@alph\c@enumii`, ma nella lista questa lettera è racchiusa fra parentesi tonde `(\theenumii)`. Nella citazione del terzo elemento non basta che appaia 'c' oppure '(c)', ma ci vuole anche un prefisso che faccia riferimento alla prima lista; ecco quindi il prefisso `\theenumi` che fa si che quell'elemento sia citato come '2(c)'. Lo stesso avviene per gli elementi delle liste interne.

Nel file `bk10.clo`, che contiene le dimensioni e i corpi specificati per quando si compone con l'opzione di default *10pt*, sono definiti alcuni dei parametri descritti per l'ambiente *list*. Per l'enumerazione di primo livello il tutto è contenuto nella definizione del comando `\@listi`; questi spazi si riferiscono a tutte le liste di primo livello, non solo alle enumerazioni; precisamente sono definiti `\parsep`,

`\topsep` e `\itemsep`; sono tutte lunghezze elastiche, perché contengono la componente di allungamento, introdotta da `\@plus`, e la componente di accorciamento, introdotta da `\@minus`; il comando `\p@` indica una lunghezza predefinita del valore di un punto tipografico. Fra le definizioni contenute in `\@listi` compare anche `\leftmargin`, a cui viene assegnato il valore di `\leftmargini`; questo a sua volta è definito nella classe *book* come una quantità espressa in unità di misura `em`, legate, cioè, al corpo del font in uso in quel contesto (potrebbe anche essere una nota o una citazione scritta in corpo minore). Definizioni analoghe appaiono anche per i livelli interni di annidamento.

Infine viene la definizione vera e propria che ricorre all'ambiente *list* così come è contenuta in `latex.ltx`. In questa definizione per prima cosa si verifica che il livello di annidamento già raggiunto non sia superiore a tre; se così fosse si emette un messaggio di errore mediante il comando `\@toodeep`, altrimenti si incrementa il contatore del livello di annidamento e si procede alle varie definizioni; si noti che per fare riferimento ai parametri corrispondenti al livello di annidamento corrente si fa riferimento al suo nome ottenuto agglutinando il prefisso `enum` con il valore del contatore di annidamento espresso in numeri romani minuscoli; questo risultato si ottiene con i comandi primitivi `\csname` e `\endcsname`; in generale, l'utente che deve usare l'ambiente *list* per confezionare una lista speciale non ha bisogno di ricorrere a questi comandi primitivi, ma a loro modo molto avanzati; all'occorrenza può tuttavia consultare il TEXbook.

Il primo argomento dell'ambiente *list* (attivato senza ricorrere a `\begin`, ma dando direttamente il comando 'interno' `\list`) non è racchiuso fra parentesi graffe, ma è il frutto di `\expandafter` e nuovamente di `\csname` e `\endcsname` che assieme agglutinano il prefisso `label` con il valore del contatore di annidamento espresso in numeri romani minuscoli.

Il secondo argomento dice di usare il contatore `\@enumctr` e dice di comporre l'etichetta di ogni voce mediante un numero composto in bandiera giustificata a destra, autorizzando anche di fuoriuscire dal margine di sinistra mediante l'accorgimento di far precedere il comando `\llap` (*left overlap*, sovrapponi a sinistra) con un blocco di gomma elastica di lunghezza naturale nulla e allungamento e accorciamento infiniti `\hss`.

Il comando di chiusura dell'ambiente *enumerate* è reso equivalente attraverso `\let` al comando di chiusura della generica lista `\endlist`.

L'esempio appena fatto, forse, è complicato, ma in realtà delucida tante cose che non sono state descritte nei dettagli negli appositi capitoli e mostra anche alcune tecniche di programmazione a livello di comandi primitivi di TEX.

Un esempio più semplice. Si vuole comporre una ambiente simile a *description* che abbia la possibilità di definire il margine sinistro sulla base della lunghezza di una parola; potrebbe essere utile per un glossario, per esempio. Vogliamo anche che ogni chiave che introduce ogni voce sia scritta in carattere lineare, ma non sia nero. Vediamo prima come è definito l'ambiente *description*:

```
\newenvironment{description}
               {\list{}{\labelwidth\z@ \itemindent-\leftmargin
```

```
                    \let\makelabel\descriptionlabel}}
              {\endlist}
\newcommand*\descriptionlabel[1]{\hspace\labelsep
                                 \normalfont\bfseries #1}
```

Dobbiamo quindi definire comandi simili, per esempio l'ambiente *descrizione*, che accetti un argomento sul quale prendere le misure del margine sinistro, e la sua etichetta `\etichettadescrizione`. Ecco come fare:

```
\newenvironment{descrizione}[1]%
      {\list{}{\settowidth{\labelwidth}{\normalfont\sffamily#1}%
                  \let\makelabel\etichettadescrizione}}
               {\endlist}
\newcommand*{\etichettadescrizione}[1]{\hspace\labelsep
                  \normalfonnt\sffamily #1}
```

Come si vede il compito era inizialmente più facile e la soluzione è decisamente più semplice.

29.5.3 Testo composto verbatim

Si possono usare due comandi e due ambienti per comporre del testo in modo verbatim; questo modo di comporre è quello che serve quando bisogna esporre, per esempio, dei brani scritti in un linguaggio di programmazione nel quale si fa uso dei caratteri speciali di TeX (ovviamente con altri significati). I comandi sono

`\verb`*simbolo*⟨*testo da riprodurre*⟩*simbolo* oppure `\verb*`*simbolo*⟨*testo da riprodurre*⟩*simbolo*

Il *simbolo* è un solo carattere e funziona da delimitatore del ⟨*testo da riprodurre*⟩; la versione senza asterisco riproduce gli spazi come spazi; la versione con l'asterisco riproduce gli spazi con il segno speciale ␣. Il ⟨*testo da riprodurre*⟩ deve comparire in una sola riga nel file sorgente.

Gli ambienti sono invece:

`\begin{verbatim}`⟨*testo da riprodurre verbatim*⟩`\end{verbatim}` `\begin{verbatim*}`⟨*testo da riprodurre verbatim*⟩`\end{verbatim*}`

Il ⟨*testo da riprodurre verbatim*⟩ si può svolgere su diverse righe, anzi di solito è composto di diverse righe. L'unica riga che non si può riprodurre è `\end{verbatim}` con o senza asterisco. L'ambiente asteriscato, come il comando asteriscato, riproduce lo spazio in modo visibile con il carattere ␣.

Il comando `\verb` è fragilissimo e non può apparire come argomento di nessun altro comando (nemmeno nella versione asteriscata).

Il pacchetto standard *alltt* definisce un ambiente *alltt* da usare come l'ambiente *verbatim* dove tutti i caratteri sono diventati caratteri 'normali', salvo `\`, `{` e `}`.

29.6 Formule matematiche

29.6.1 Formule

La matematica in linea può essere delimitata dai seguenti delimitatori:

```
$ ⟨formula in linea⟩ $
\( ⟨formula in linea⟩ \)
\begin{math} ⟨formula in linea⟩ \end{math}
```

Si consiglia di non usare il primo metodo, anche se è più comodo da scrivere, perché si perde la diagnostica di LaTeX nel caso che ci si dimentichi di uno dei due delimitatori. L'ambiente *math*, come si può ben capire, non viene usato molto spesso. I delimitatori `\(` e `\)` sono i migliori, ma sono fragili. Invece i delimitatori `$`...`$` sono robusti.

Il comando

```
\ensuremath{⟨formula⟩}
```

permette di comporre una ⟨*formula*⟩ in linea garantendo che essa venga composta correttamente anche se il comando viene emesso in modo testuale. È comodo per definire macro che debbono potersi usare sia nel modo testo sia nel modo matematico.

Usando il pacchetto `babel` è disponibile il comando:

```
\textormath⟨per il modo testo⟩⟨per il modo matematico⟩
```

che produce un effetto analogo a quello di `\ensuremath`, con una notevole differenza; usato per definire un comando da usare sia in modo testo sia in modo matematico, può eseguire composizioni diverse nell'uno o nell'altro modo senza che il compositore debba preoccuparsi di verificare il modo di composizione; `\ensuremath`, invece, compone il suo argomento sempre in modo matematico, all'occorrenza entrando ed uscendo da questo modo, al fine di lasciare TeX nello stesso stato nel quale si trovava prima dell'esecuzione del comando.

Per la matematica in display si hanno gli ambienti

```
\[ ⟨formula in display⟩ \]
\begin{displaymath} ⟨formula in display⟩ \end{displaymath}
\begin{equation} ⟨formula in display⟩ \end{equation}
```

I primi due ambienti non assegnano un numero alla ⟨*formula in display*⟩, mentre il terzo ambiente assegna un numero a detta formula. L'ambiente *displaymath* viene usato raramente visto che quello definito con `\[`...`\]` è molto più comodo da usare. L'ambiente *equation* non ha sostituti.

LaTeX offrirebbe anche gli ambienti *eqnarray* e *eqnarray** ma, per i motivi già esposti nel capitolo 14, è preferibile non servirsene; al contrario per comporre

formule e sistemi di formule in display è molto meglio servirsi del pacchetto *amsmath*.

Dentro agli ambienti *equation* e *eqnarray* (all'interno di ciascuna riga) è possibile inserire il comando `\label` così da definire una etichetta mnemonica per richiamare il numero della formula con `\ref` e/o con `\pageref`. Dentro *eqnarray* si può usare `\nonumber` per evitare che LaTeX assegni un numero ad una espressione che non si vuole numerare. Questo ambiente consentirebbe anche di spezzare una singola lunga espressione matematica su più righe, ricorrendo anche al comando suddetto e a `\lefteqn`; ma per questo scopo è meglio servirsi degli opportuni ambienti predisposti apposta dal pacchetto *amsmath*.

Le espressioni in display usano un certo numero di parametri per comporre secondo diversi stili.

`\jot` serve per aggiungere altro spazio fra una equazione e l'altra in un sistema di equazioni.

`\mathindent` serve per fissare la rientranza sinistra delle equazioni quando queste sono composte allineate a sinistra se si usa l'opzione *fleqn* per la classe del documento.

`\abovedisplayskip` serve per definire il contrografismo che precede una equazione in display; viene usato un contrografismo più piccolo quando la riga prima del display è molto corta e l'espressione matematica potrebbe dare l'impressione ottica di essere preceduta da uno spazio troppo grande.

`\abovedisplayshortskip` è appunto questo contrografismo superiore più piccolo.

`\belowdisplayskip` è il contrografismo da inserire dopo una formula per distanziarla dal testo seguente.

`\belowdisplayshortskip` serve per inserire un contrografismo più piccolo se l'espressione matematica è seguita da una riga di testo molto corta.

Va da sé che i `...shortskip` non vengono usati quando si compone con l'opzione *fleqn* perché le espressioni sono tutte allineate a sinistra, a meno del loro rientro non molto importante (una cinquantina di punti, circa 15 mm).

Per comporre le espressioni matematiche si usano spesso piccoli comandi o piccole strutture di cui è facile dimenticarsi.

Esponenti gli esponenti si inseriscono mediante il comando `^`

`^{`⟨*esponente*⟩`}`

Con l'opzione *italian* di *babel* è disponibile il comando `\ap{`⟨*apice*⟩`}` per inserire un apice in tondo quando si è in modo matematico, un apice con

il font corrente quando si è in modo testo. Il nucleo di LaTeX per il modo testo prevede anche il comando `\textsuperscript`, valido sempre, anche senza l'uso di *babel*.

Apici L'apice ′ o il doppio apice ″ si inseriscono con uno o due apostrofi, che in modo matematico vengono composti come apici.

Pedici e deponenti I pedici vengono inseriti con il comando `_`

`_{`⟨*pedice*⟩`}`

Con l'opzione *italian* di *babel* è possibile usare il comando `\ped{`⟨*pedice*⟩`}` per inserire un pedice in tondo in modo matematico e per inserire un deponente con il font corrente in modo testo.

Frazioni Le frazioni vengono composte con il comando

`\frac{`⟨*numeratore*⟩`}{`⟨*denominatore*⟩`}`

Bisogna ricordare che questo comando compone le frazioni in display come ci si aspetterebbe, ma le compone in modo testo sia nelle espressioni matematiche in linea, sia nelle sottoespressioni come per esempio un altro numeratore o denominatore; in questi casi è preferibile servirsi delle frazioni a barra obliqua, piuttosto che di quelle a barra orizzontale.

Radici Le radici quadrate o con altro ⟨*indice*⟩ si compongono con

`\sqrt[`⟨*indice*⟩`]{`⟨*radicando*⟩`}`

Se l'⟨*indice*⟩ vale 2 non lo si indica.

Ellissi Le parti omesse (ellissi) vengono sostituite con i soliti tre puntini; in matematica ce ne hanno di diversi tipi: `\dots` (…) si può usare sia in modo matematico sia in modo testo e produce i soliti puntini a livello della linea di base in modo testo, o a livello della linea di base o dell'asse matematico in modo matematico a seconda del contesto, cioè se è preceduto o seguito da segni di interpunzione appoggiati alla linea di base oppure di operatori binari centrati sull'asse matematico; `\ldots` produce lo stesso effetto di collocare i tre puntini sulla linea di base, ma è da usare in modo matematico; `\cdots` (⋯) produce i tre puntini allineati con il segno −, cioè sull'asse matematico della formula; `\vdots` (⋮) produce tre puntini verticali che servono per sostituire in verticale gli elementi omessi da colonne di matrici; infine `\ddots` (⋱) produce tre puntini in diagonale, utili per riempire una parte omessa dal cuore di una matrice.

Se si usa `\dots` in modo testo i tre puntini sono sempre sulla linea di base; ma se si usa il pacchetto *amsmath*, questo comando è ridefinito

in modo che in matematica esso si comporti in modo intelligente, vale a dire esso colloca i puntini sulla linea di base se almeno da un lato è contornato da un segno di interpunzione, mentre esso produce i puntini sull'asse matematico della formula se è contornato da almeno un operatore binario. Si confrontino i seguenti due casi, composti entrambi con `\dots`:

$$i = 1, 2, 3, \ldots, n \qquad i = 1 + 2 + 3 + \cdots + n$$

29.6.2 Simboli, accenti, delimitatori e grandi operatori

Tutti i simboli di qualunque genere usabili in matematica sono raccolti nelle tabelle 13.2–13.8; nelle tabelle 14.1–14.2, invece, sono raccolti gli ulteriori simboli disponibili con il pacchetto *amsmath*.

29.6.3 Impilare gli oggetti matematici

Una (breve) espressione può venire soprallineata con

`\overline{`⟨*espressione*⟩`}`

Analogamente si può sottolineare una espressione con

`\underline{`⟨*espressione*⟩`}`

Anche gli accenti matematici possono venire impilati, ma si tratta semplicemente di racchiudere fra le graffe che delimitano l'argomento del primo accento una espressione già accentata.

Invece `\stackrel` permette di costruire degli operatori di relazione impilando uno sopra l'altro due segni distinti, scelti fra i vari simboli disponibili:

`\stackrel{`⟨*elemento superiore*⟩`}{`⟨*elemento inferiore*⟩`}`

29.6.4 Spaziatura matematica

Non si può raccomandare mai abbastanza l'opportunità di non spaziare le espressioni matematiche. I comandi di spaziatura hanno senso se e solo se la spaziatura di default non è adeguata ai simboli che compaiono uno accanto all'altro in una specifica espressione matematica. Se, quindi, si inserirà una qualche spaziatura, lo si farà solo dopo aver corretto le bozze controllando accuratamente che questa spaziatura sia praticamente impercettibile.

`\,`	spazio sottile	`\:`	spazio medio
`\!`	spazio sottile negativo	`\;`	spazio grande

Il comando `\,` può essere usato anche in modo testo. Naturalmente si potrebbero anche usare `\quad` e `\qquad` oltre ai comandi di spaziatura del modo testo. Tuttavia merita di segnalare i comandi primitivi `\mskip` e `\mkern` che accettano

come argomenti (non racchiusi fra nessun tipo di parentesi) degli spazi espressi mediante le unità di misura chiamate `mu`, valide solo in matematica; 18 mu equivalgono a 1 em, ma in matematica l'unità 'em' cambia grandezza a seconda del font in uso, in particolare per il corpo principale della formula, per gli apici ed i pedici di primo ordine e per gli apici ed i pedici di secondo ordine.

29.6.5 Font matematici

Per avere una intera formula composta con caratteri medi o con caratteri neri bisogna specificarlo prima di entrare in modo matematico:

```
\boldmath
⟨ambiente matematico⟩
\unboldmath
```

Altrimenti i vari simboli letterali e/o gli operatori possono essere resi con font diversi se si usano i comandi seguenti:

`\mathrm`	tondo	`\mathsf`	senza grazie
`\mathit`	*corsivo*	`\mathtt`	`spaziatura fissa`
`\mathbf`	**nero**	`\mathcal`	$\mathcal{CALLIGRAFICO}$

Si noti che il corsivo matematico, il corsivo testuale usato in matematica mediante il comando `\mathit`, e il corsivo per il testo sono font con proprietà completamente diverse; per esempio il primo non contiene legature, per cui la parola *affine* verrebbe composta $affine$. Più dettagliatamente gli esempi seguenti mostrano il diverso comportamento dei tre corsivi.

`\textit{affine, affine affine}`	*affine, affine affine*
`\( \mathit{affine, affine affine} \)`	$affine, affineaffine$
`\( affine, affine affine \)`	$affine, affineaffine$

Si noti che i comandi testuali, come `\textit`, possono essere usati anche in matematica producendo semplice testo; mentre i comandi che scelgono i font in matematica lo fanno solo per le lettere e gli altri segni equivalenti alle lettere. Il corsivo matematico trae i suoi segni da una polizza diversa da quella sia del testo corsivo sia del testo corsivo matematico. Si noti ancora che lo spazio dopo la virgola, quando si scrive usando il comando `\mathit`, appare esplicitamente, perché nel contesto di questo libro si è usata la 'virgola intelligente', vale a dire si è usato il significato speciale di carattere attivo che la virgola assume in matematica al fine di inserire uno spazio se seguita da uno spazio, ma di non inserire nulla se è seguita da un altro segno; si nota infatti che la virgola sia nella seconda sia nella terza riga è seguita da uno spazio sottile (matematico), mentre nella prima riga essa è seguita da un normale spazio interparola.

29.6.6 Stili di composizione

I quattro stili di composizione della matematica sono

```
\displaystyle
\textstyle
\scriptstyle
\scriptscriptstyle
```

Lo stile `\textstyle` differisce dal `\displaystyle` nel senso che è più raccolto in verticale: gli apici e i pedici sono più vicini all'asse matematico; i limiti superiori e inferiori sono composti accanto all'operatore come se fossero apici o pedici; le frazioni sono composte in `\scriptstyle` per mantenere limitato l'ingombro verticale in modo che non sia necessario spaziare le righe del testo. È per questo che bisogna stare attenti, specialmente con le frazioni, a non renderle troppo piccole; piuttosto che una frazione troppo piccola, è preferibile una barra obliqua, senza rimpicciolire numeratori e denominatori, nemmeno quando sono puramente numerici, come in $^2\!/_3$; questa pratica è 'deprecata' dalla norma UNI 2950.

29.7 Definizioni, numeri e programmazione

29.7.1 Comandi di definizione

Nuove macro possono essere definite, ridefinite o rese disponibili, se non lo sono già, mediante i comandi:

```
\newcommand{⟨comando⟩}[⟨numero argomenti⟩][⟨default⟩]{⟨definizione⟩}
\renewcommand{⟨comando⟩}[⟨numero argomenti⟩][⟨default⟩]{⟨definizione⟩}
\providecommand{⟨comando⟩}[⟨numero argomenti⟩][⟨default⟩]{⟨definizione⟩}
```

Sono tutti comandi fragili, ma del resto non sembra opportuno usarli negli argomenti di altri comandi. Sono già stati descritti in dettaglio nel capitolo 19. Tutti dispongono della versione asteriscata; la differenza sembra minima, ma è importante per evitare perdite di tempo quando si commettono errori nell'uso dei comandi definiti mediante queste istruzioni. In termini di gergo TeX, i comandi senza asterisco sono `\long`, mentre quelli con asterisco non lo sono. Un 'long command' che accetta uno o più argomenti è in grado di elaborare anche argomenti composti di più capoversi; per esempio `\parbox` è un 'long command'. I comandi che non hanno questo attributo non consentono che i loro argomenti contengano una linea bianca e/o un comando esplicito di fine capoverso, `\par`; se questo succedesse essi arresterebbero subito la compilazione con un messaggio di errore. Questo fatto succede abbastanza frequentemente quando ci si dimentica la parentesi graffa di chiusura di un argomento. Ma con i comandi 'corti' l'errore viene trovato subito con la normale diagnostica di LaTeX, mentre con i comandi lunghi l'errore potrebbe venire segnalato o alla fine del file oppure perché la memoria di TeX è satura.

29.7.2 Comandi per la definizione di ambienti

Gli ambienti sono definiti o ridefiniti mediante i comandi

```
\newenvironment{⟨nome⟩}[⟨numero argomenti⟩][⟨default⟩]%
{⟨comandi di apertura⟩}{⟨comandi di chiusura⟩}
\renewenvironment{⟨nome⟩}[⟨numero argomenti⟩][⟨default⟩]%
{⟨comandi di apertura⟩}{⟨comandi di chiusura⟩}
```

Si veda il capitolo 19 per maggiori dettagli ed esempi.

29.7.3 Teoremi

LaTeX mette a disposizione dei comandi per definire degli pseudo ambienti al fine di comporre gli enunciati di teoremi o enunciati simili. Il comando `\newtheorem` consente due forme di definizione:

```
\newtheorem{⟨nome⟩}{⟨titolino⟩}[⟨contatore dominante⟩]
\newtheorem{⟨nome⟩}[⟨numerato come⟩]{⟨titolino⟩}
```

dove:

⟨*nome*⟩ rappresenta il nome dello pseudoambiente; potrà essere, per esempio, teorema, lemma, definizione, eccetera. Per cui, se ⟨*nome*⟩ si riferisce a un `teorema`, il suo enunciato verrà racchiuso all'interno di `\begin{teorema}` ⟨*enunciato*⟩ `\end{teorema}`.

⟨*titolino*⟩ la parola che identifica l'enunciato; potrà essere Teorema, Lemma, Definizione, eccetera. Non si confonda il nome dell'ambiente con il titolino dell'enunciato.

⟨*contatore dominante*⟩ è il contatore del capitolo, se si vuole che la numerazione del nuovo enunciato sia iniziata da 1 ad ogni nuovo capitolo; sarà il contatore dei paragrafi, se si desidera che la numerazione ricominci da 1 ad ogni nuovo paragrafo; eccetera.

⟨*numerato come*⟩ è invece il nome dello pseudo ambiente del quale si vuole condividere la numerazione; Questo deve essere già stato definito. Si potrebbe per esempio numerare con una sola sequenza numerica sia i teoremi sia i lemmi sia i corollari, mentre, probabilmente, si preferirebbe numerare separatamente le definizioni, le congetture, eccetera.

Se non si specificano i contatori facoltativi, questi comandi definiscono un nuovo contatore con lo stesso nome dello pseudo ambiente che può venire stampato usando il comando `\the`*nome*. Nella stessa maniera i comandi `\label` e `\ref` usano il nuovo nome del contatore per rendere accessibile ai riferimenti incrociati anche questi enunciati. Se invece si specifica il contatore ⟨*numerato come*⟩ non viene introdotto nessun nuovo contatore ma si usa quello già esistente. Infine se

si specifica il nome del ⟨*contatore dominante*⟩, il comando `\the`*nome* produrrà in stampa il numero del contatore relativo allo pseudo ambiente preceduto dal numero stampato del ⟨*contatore dominante*⟩; se per esempio il ⟨*contatore dominante*⟩ fosse il paragrafo, allora il tredicesimo enunciato del quarto paragrafo del decimo capitolo verrà stampato nella forma 10.4.13.

29.8 Numeri, lunghezze e spazi

Il sistema TeX per comporre svolge una quantità di calcoli sia usando numeri veri e propri, sia usando lunghezze e spazi.

TeX fa calcoli solo con numeri interi; gli unici numeri non interi che esso gestisce sono i fattori moltiplicativi delle lunghezze o degli spazi. Qui di seguito vedremo come esistano registri per conservare i numeri, come agire sui numeri e sui loro registri, chiamati *contatori*.

TeX usa le lunghezze e dispone di registri per la conservazione di informazioni metriche; questi registri non hanno nomi particolari, come i contatori per i numeri, ma possono essere *chiamati per nome*, come lo si può fare per i contatori.

Gli spazi sono speciali lunghezze e i loro registri sono anch'essi un po' speciali; essi sono chiamati anche "lunghezze elastiche", perché hanno la proprietà di potersi allungare o accorciare rispetto alla loro lunghezza naturale; essi sono formati dalla collezione di tre lunghezze, la prima rappresenta la lunghezza naturale, la seconda l'ammontare assoluto di cui essa può allungarsi, e la terza l'ammontare assoluto con cui può restringersi.

Per operare con i numeri e le lunghezze rigide o elastiche TeX dispone di un certo numero di comandi primitivi. L'estensione `etex` incorporata nelle versioni più recenti di `pdftex`, consentono di eseguire certe operazioni in modo molto più comodo. Altrimenti, non disponendo di una versione recente di `pdftex`, bisogna ricorrere al pacchetto *`calc`*, alla cui documentazione si rimanda il lettore.

29.8.1 Numeri

A parte il modo di scrivere i numeri già visto nella pagina 642, LaTeX consente di fare semplici conti con numeri interi e/o con le misure delle lunghezze.

`\newcounter` con la sintassi

`\newcounter{`⟨*nome*⟩`}[`⟨*contatore dominante*⟩`]`

permette di definire un nuovo contatore per numeri interi chiamato ⟨*nome*⟩, asservito al ⟨*contatore dominante*⟩; quindi viene azzerato, quando il ⟨*contatore dominante*⟩ viene incrementato. Il ⟨*nome*⟩ e il ⟨*contatore dominante*⟩ hanno lo stesso significato descritto per i teoremi.

`\setcounter` con la sintassi

`\setcounter{`⟨*nome*⟩`}{`⟨*valore*⟩`}`

inserisce la quantità numerica ⟨*valore*⟩ nel contatore che ha quel ⟨*nome*⟩.

`\addtocounter` con la sintassi

`\addtocounter{`⟨*nome*⟩`}{`⟨*valore*⟩`}`

aggiunge il ⟨*valore*⟩ specificato al contenuto del contatore ⟨*nome*⟩. Il ⟨*valore*⟩ può anche essere negativo, quindi di fatto LaTeX esegue una somma algebrica.

`\value` con la sintassi

`\value{`⟨*nome*⟩`}`

recupera il contenuto del contatore ⟨*nome*⟩ per passarlo ad altri comandi o per rendere stampabile il numero contenuto dentro quel contatore.

`\stepcounter` con la sintassi

`\stepcounter{`⟨*nome*⟩`}`

incrementa il contatore ⟨*nome*⟩ di una unità.

`\refstepcounter` con la sintassi

`\refstepcounter{`⟨*nome*⟩`}`

oltre ad incrementare il contatore ⟨*nome*⟩ di una unità lo rende accessibile al comando `\label` così che si possa fare riferimento ai valori del contatore mediante le chiavi usate da `\label` e `\ref`.

`\arabic` con la sintassi

`\arabic{`⟨*nome*⟩`}`

trasforma il contenuto del contatore ⟨*nome*⟩ nell'equivalente stringa di cifre arabe.

`\roman` con la sintassi

`\roman{`⟨*nome*⟩`}`

trasforma il contenuto del contatore ⟨*nome*⟩ nell'equivalente stringa di cifre romane minuscole; nel testo si è già commentato a proposito dell'infelice scelta delle cifre tonde minuscole, mentre sarebbe meglio ricorrere sempre al cifre minuscole del maiuscoletto; questo però non è disponibile in tutte le famiglie, serie e forme, quindi l'utente deve provvedere a mano caso per caso.

`\Roman` con la sintassi

```
\Roman{⟨nome⟩}
```

trasforma il contenuto del contatore ⟨*nome*⟩ nell'equivalente stringa di cifre romane maiuscole.

`\alph` con la sintassi

```
\alph{⟨nome⟩}
```

trasforma il contenuto del contatore ⟨*nome*⟩ nella lettera dell'alfabeto latino minuscolo corrispondente alla posizione indicata dal contatore; ovviamente in questo modo si può rappresentare il valore del contatore solo finché il suo contenuto è positivo ma non superiore a 26, essendo 26 le lettere dell'alfabeto latino.

`\Alph` con la sintassi

```
\Alph{⟨nome⟩}
```

trasforma il contenuto del contatore ⟨*nome*⟩ nella lettera dell'alfabeto latino maiuscolo corrispondente alla posizione indicata dal contatore; ovviamente in questo modo si può rappresentare il valore del contatore solo finché il suo contenuto è positivo ma non superiore a 26, essendo 26 le lettere dell'alfabeto latino.

`\fnsymbol` LaTeX contiene una lista ordinata di simboli, ∗, †, ‡, §, ¶, ‖, ∗∗, ††, ‡‡, dalla quale il comando `\fnsymbol`, con la sintassi

```
\fnsymbol{⟨nome⟩}
```

trae il simbolo corrispondente alla posizione indicata dal contatore ⟨*nome*⟩; questo comando serve solo per i richiami di nota del comando `\thanks`, che può essere usato solo per comporre la lista degli autori, il titolo e la data quando si usa il comando `\maketitle` o quando si usa l'ambiente *titlepage*. Indicando le note con questi simboli si possono rendere univoche le note con un numero positivo, non superiore a 9, visto che la lista contiene solo nove simboli.

Quando si definisce un nuovo contatore ⟨*nome*⟩ LaTeX definisce anche il modo di stamparlo attraverso il comando `\the`⟨*nome*⟩; In altre parole se si definisce il nuovo contatore `pippo` mediante il comando:

```
\newcounter{pippo}
```

viene contemporaneamente definito il comando `\thepippo`, in generale in modo molto semplice[1]:

```
\newcommand\thepippo{\arabic{pippo}}
```

Il compositore è libero di ridefinire il modo di scrivere il contatore, specialmente se è asservito ad un altro contatore; a titolo di esempio scriverà:

```
\newcounter{pippo}[pluto]
\renewcommand*\thepippo{\arabic{pluto}.\arabic{pippo}}
```

se vuole che il contatore `pippo`, supponendo che valga 3, quando il contatore `pluto` vale 5 venga stampato nella forma 5.3.

Vale la pena di segnalare che il comando primitivo `\the` può precedere il nome di qualunque registro interno dell'interprete per trasformarne il contenuto in una stringa di caratteri che ne indica il contenuto. Questo comando primitivo non è applicabile ai nomi dei contatori LaTeX, perché i loro nomi non individuano direttamente il registro, come invece succede con le lunghezze. Quando si definisce un contatore LaTeX, viene associato un particolare registro numerico, il primo disponibile, ad una sequenza formata con il ⟨*nome*⟩ preceduto alla stringa `\c@`. Quindi il nome del contatore LaTeX `pippo` serve per definire la sequenza `\c@pippo` che viene associata all'indirizzo numerico del registro interno (di cui il compositore non deve preoccuparsi). Il comando `\the` dovrebbe quindi precedere la sequenza `\c@pippo` così:

```
\the\c@pippo
```

per produrne la stringa di cifre che ne indica il contenuto. Data la presenza del segno `@`, che normalmente non è una lettera, anche se può essere usato come lettera all'interno dei file di classe o di quelli delle estensioni, questo modo di procedere è fortemente sconsigliato; tra l'altro è molto più comodo, e meno incline agli errori, ricorrere ai comandi specifici, come per esempio `\arabic`, per scrivere il contenuto dei contatori LaTeX; in ogni caso il valore numerico contenuto dentro un registro, indipendentemente da modo di scriverlo può sempre essere usato tramite l'intermediario del comando `\value`: `\value{pippo}` restituisce il valore numerico contenuto dentro il contatore LaTeX `pippo`, e questo valore può venire usato come meglio si crede, anche come argomento dei test numerici `\ifnum`, `\ifodd` e simili.

[1] In realtà la definizione normalmente viene eseguita ricorrendo ai comandi primitivi che sono più veloci da eseguire in quanto non hanno bisogno di essere interpretati.

29.8.2 Lunghezze

Le lunghezze esplicite, quelle, cioè, che si esprimono con una misura e una unità di misura, sono costituite da una sequenza di cifre decimali, separate dal separatore decimale che in teoria potrebbe anche essere la virgola; siccome nella sintassi LaTeX la virgola svolge anche altri ruoli, è preferibile usare sempre e soltanto il punto decimale quando si esprimono le misure delle lunghezze. Un numero che contenga una parte intera nulla potrebbe cominciare con il punto decimale, omettendo lo zero che dovrebbe precederlo; si sconsiglia questa abbreviazione del tutto irrilevante ma che facilita gli errori. Il numero può essere preceduto dal segno positivo o negativo; quello positivo è evidentemente facoltativo; quello negativo invece non lo è. Se in seguito alla sostituzione di espressioni ed argomenti nelle macro la misura risultasse preceduta da diversi segni, il numero di segni negativi si comporta come ci si aspetterebbe, nel senso che ognuno cambia segno a quanto segue, quindi un numero dispari di segni negativi porta ad un risultato complessivamente negativo, mentre un numero pari porta ad un risultato positivo.

Le lunghezze definite con i comandi LaTeX sono sempre lunghezze elastiche, anche se i loro allungamenti e accorciamenti sono nulli; di fatto si comportano come lunghezze rigide, a meno che nella definizione non vengano specificati anche l'allungamento e/o l'accorciamento.

Unità di misura Le unità di misura accettabili da LaTeX sono le seguenti:

- `cm` centimetri.
- `mm` millimetri.
- `in` pollici.
- `pt` punti tipografici (1 pt = (1/72,27) in = 0,3515 mm).
- `pc` pica (1 pc = 12 pt).
- `bp` punto PostScript o *big point* (1 bp = (1/72) in = 0,3528 mm).
- `sp` *scaled point* (la frazione 2^{-16} di un punto tipografico).
- `dd` punto didot (1 dd=0,3760 mm=1,070 pt).
- `cc` cicero (1 cc=12 dd).
- `ex` *x-height*, occhio del carattere corrente, altezza della lettera 'x'.
- `em` *em-width*, larghezza della 'M', convenzionalmente circa uguale al corpo del font corrente.

`\fill` è una lunghezza elastica (gomma) di lunghezza naturale nulla ma infinitamente allungabile.

`\stretch{`⟨*moltiplicatore*⟩`}` è una lunghezza elastica (gomma) di lunghezza naturale nulla ma allungabile infinitamente quanto `\fill` moltiplicata per ⟨*moltiplicatore*⟩; questo è un numero decimale con segno facoltativo; se il segno è negativo questa lunghezza diventa 'infinitamente' accorciabile.

`\newlength{`⟨*comando*⟩`}` definisce una nuova lunghezza identificata con ⟨*comando*⟩; questo è il nome di un tipico comando LaTeX formato da una stringa letterale preceduta dal backslash, oppure da un solo segno non letterale preceduto dal backslash. Si possono definire fino a 256 lunghezze; in realtà il motore di composizione pdftex (sufficientemente recente) accetta la definizione di un numero molto maggiore di lunghezze, ma per compatibilità con il passato fino ad oggi (2010) non si è ancora visto nessun pacchetto di estensione di una certa importanza farne uso.

`\setlength{`⟨*comando*⟩`}{`⟨*lunghezza*⟩`}` serve per assegnare la ⟨*lunghezza*⟩ esplicita al registro-lunghezza identificato con ⟨*comando*⟩.

`\addtolength`⟨*comando*⟩⟨*lunghezza*⟩ serve per incrementare il valore di lunghezza contenuto nel registro ⟨*comando*⟩ della quantità ⟨*lunghezza*⟩, che può essere sia positiva, sia negativa; in questo caso si ha una sottrazione.

`\settowidth`, `\settoheight` e `\settodepth` seguono la sintassi seguente:

```
\settowidth{⟨comando⟩}{⟨testo⟩}
\settoheight{⟨comando⟩}{⟨testo⟩}
\settodepth{⟨comando⟩}{⟨testo⟩}
```

Ognuna di queste istruzioni assegna al registro-lunghezza ⟨*comando*⟩ rispettivamente la larghezza, l'altezza o la profondità della stringa che costituisce il ⟨*testo*⟩.

29.8.3 Operazioni fra numeri e grandezze

Le recenti estensioni del programma pdftex includono tutte le estensioni del precedente etex; una descrizione esaustiva è contenuta nel documento `/textmf-distr/doc/etex/base/etex_man.pdf`. quello che ora ci interessa di più è la sezione 3.5 di quel documento intitolata "Expressions".

Vediamo dunque che ora pdftex è in grado di calcolare alcune espressioni matematiche direttamente in memoria (nei registri della CPU del calcolatore che si sta usando). Il risultato di queste espressioni è usabile in qualunque momento sia lecito usare un risultato numerico o dimensionale, per esempio assegnandolo ad un registro o inserendolo in una espressione logica.

Le espressioni dimensionali `\dimexpr` e quelle numeriche `\numexpr` possono venire mescolate; non è sempre obbligatorio terminare ogni espressione di ciascun tipo con `\relax` ma è consigliabile. Esistono espressioni anche per le lunghezze elastiche e per le lunghezze matematiche: `\glueexpr` e `\muexpr`; si possono usare le parentesi tonde. L'esempio che il testo propone è il seguente:

```
\ifdim\dimexpr (2pt-5pt)*\numexpr 3-3*13/5\relax + 34pt/2 < \wd20
```

Vi compare una espressione mista dimensionale che contiene al suo interno una espressione numerica; il comando `\relax` serve per terminare l'espressione numerica; quella dimensionale termina con il primo carattere non valido in una espressione, vale a dire che termina subito prima del segno di minore `<`.

Le operazioni valide all'interno delle espressione sono i segni delle quattro operazioni aritmetiche: `+`, `-`, `*`, e `/`

Le operazioni numeriche sono eseguite fra numeri interi (o con l'indicazione di contatori TeX oppure contatori LaTeX passati come argomento di `\value`). le divisioni, quindi, sono divisioni intere, cioè forniscono solo la parte intera del quoziente che potremmo calcolare con una calcolatrice ordinaria (o anche a mano).

Le operazioni dimensionali sono le stesse e sono del tutto equivalenti a quelle che potrebbero venire eseguite usando i comandi primitivi di pdftex, vale a dire `\advance` per la somma algebrica, `\multiply` per la moltiplicazione e `\divide` per la divisione intera.

C'è però una novità: le operazioni di scalamento di una lunghezza sono intese come il prodotto della lunghezza da scalare per il numeratore del fattore di scala con il risultato diviso per il denominatore del fattore di scala; numeratore e denominatore possono essere numeri interi (eventualmente contenuti in contatori) oppure altre due lunghezze; non possono essere uno un numero e l'altro una lunghezza, ma devono essere entrambi dello stesso tipo.

Bene questa operazione di scalamento viene fatta mettendo il risultato della prima moltiplicazione in un registro della CPU a 64 bit, il doppio di una normale parola di 4 byte, e il risultato della successiva divisione viene riportato ad una parola di 32 bit, cioè di 4 byte. Nel fare questo si perdono dei bit nella parte meno significativa della misura della lunghezza, e pdftex non esegue un semplice troncamento, ma esegue l'arrotondamento degli *sced points* al valore intero più vicino e quindi il valore in punti tipografici alla quinta cifra decimale più vicina.

Vediamo un esempio: se la giustezza `\textwidth` dovesse venire divisa per sei al fine di comporre a sei colonne, bisogna togliere cinque volte lo spazio intercolonna e dividere il risultato per sei; il tutto va poi assegnato alla lunghezza `\columnwidth`; scriveremo allora[2]:

```
\newlength\Numer \newlength\Denom
...
\setlength\Numer{1pt} \setlength\Denom{6pt}
\columnwidth=\dimexpr(\textwidth -5\columnsep)*\Numer/\Denom\relax
```

Si noti che moltiplicare per 1 pt e dividere per 6 pt non è la stessa cosa di moltiplicare per 1 e dividere per 6. Nella codifica interna la lunghezza di 1 pt è data dal numero intero di *scaled points* corrispondenti appunto a 1 pt, quindi da un numero binario formato da un 1 seguito da sedici zeri binari; analogamente il numero sei è rappresento da tre cifre binarie `110`, seguite da sedici zeri binari; la prima moltiplicazione pertanto sposta a sinistra di sedici posizioni il

[2] Usando la sintassi di basso livello di pdftex.

numero del risultato della prima sottoespressione, poi viene diviso per sei e il risultato viene scalato a destra di sedici posizioni, portando ad un suo eventuale arrotondamento. Moltiplicare semplicemente per uno e dividere semplicemente per sei non porta con sé nessuno scalamento e nessun arrotondamento. Infatti eseguendo l'operazione indicata sopra l'espressione calcolata con `\dimexpr` vale 49,16667 pt, mentre quella calcolata con gli operatori primitivi vale 49,16666 pt. In questo caso la differenza è trascurabile, ma in altri casi potrebbe non esserlo.

Si noti infine che le operazioni di scalamento vengono eseguite correttamente se si usano i registri lunghezza (o i registri numerici) piuttosto che le grandezze esplicite.

Vale la pena di indicare alcune particolarità dei calcoli che pdftex esegue e dei registri in cui salva certe quantità; è importante anche al fine di rendersi conto della delicatezza dei calcoli e dell'eventualità di eseguirne alcuni che portano o alla perdita di ogni informazione o all'*overflow*.

I registri interi, cioè i contatori sono parole di 4 byte, che contengono 32 bit o cifre binarie. Un bit è riservato per il segno, quindi restano 31 bit per rappresentare i numeri. Ne segue che il massimo numero intero vale $2^{31} - 1 = 2\,147\,483\,647$; chiaramente un numero sufficientemente grande da dubitare di poterlo superare.

I registri per le lunghezze sono ugualmente dei registri interi dove viene memorizzato il numero di *scaled points* corrispondenti. Si tratta di parole di 4 byte che contengono 32 bit; due di questi bit servono per il segno e per altre informazioni specifiche relative alle lunghezze; restano quindi 30 bit per memorizzare la grandezza espressa mediante un numero intero di *scaled points* il cui valore assoluto massimo può quindi arrivare a 1 073 741 823. Siccome 2^{16} sp corrispondono a 1 pt, la massima lunghezza espressa in punti che può venire memorizzata vale 16 383,99999 pt. Questo valore, tra l'altro viene memorizzato nel registro di lunghezza `\maxdimen`, nel caso il compositore/programmatore volesse eseguire dei confronti o utilizzare questo numero speciale.

Questa massima lunghezza corrisponde a poco più di 5758 mm, quasi sei metri e sembra che anche questo valore in tipografia sia difficilmente raggiungibile. Purtroppo non è vero Si pensi infatti ai calcoli che bisognerebbe fare per determinare una lunghezza che sia in proporzione con un'altra lunghezza con lo stesso rapporto di altre due lunghezze:

$$\frac{l_1}{l_2} = \frac{L_1}{L_2} \qquad \text{cioè} \qquad l_1 = l_2 \, \frac{L_1}{L_2}$$

Per fare un esempio pratico, si supponga di avere a disposizione una carta di formato non standard con base L_2 e altezza L_1 da specificare nel preambolo assegnandone i valori ai registri di lunghezza `\paperwidth` e `\paperheight`; in base al font usato si determina che la leggibilità è ottimale se la base della griglia interna `\textwidth` vale l_2; quanto deve valere `\textheight` affinché il testo sia inscritto in un rettangolo simile al rettangolo della carta?

Con le espressioni di dimensione che abbiamo appena visto possiamo determinare:

```
\textheight=\dimexpr \textwidth * \paperheight / \paperwidth \relax
```

e pdftex ci esegue i calcoli senza problemi grazie ai risultati interni conservati un un registro di due parole, cioè di 64 bit.

Cosa sarebbe successo se avessimo usato i comandi primitivi `\multiply` e `\divide`? Per rendercene conto è meglio fare un esempio numerico. Supponiamo che la carta abbia le dimensioni di 220 mm per 280 mm (il vecchio formato italiano quadrotto, simile al formato letter americano, ma che in Italia non è più facilmente reperibile, per non dire che è irreperibile). Il calcolo della giustezza sulla base dei font che andiamo ad usare ci dice che questa dovrebbe valere 170 mm. Eseguendo i calcoli con una calcolatrice tascabile (a virgola mobile) avremmo[3]:

$$\begin{aligned} x &= l_2 \cdot L_1 = 170 \times 280 = 47600 \gg 5758 \\ l_1 &= x / L_2 = 47600/220 = 216{,}364 \end{aligned}$$

Come si vede il risultato finale è perfettamente accettabile, ma la moltiplicazione intermedia eccede le possibilità dei registri interni di pdftex e quindi si ha un *overflow*.

Ah, se è così, allora eseguiamo prima la divisione e poi la moltiplicazione:

$$\begin{aligned} y &= l_2 / L_2 = 170/220 = 0 \\ l_1 &= y \cdot L_1 = 0 \times 280 = 0 \end{aligned}$$

ma in questo modo la divisione intera di un numero per un numero più grande ci dà la parte intera del quoziente che vale evidentemente zero e perdiamo ogni informazione.

Ecco, quindi, dove nasce l'enorme comodità dell'operazione di scalamento eseguita da pdftex in un registro interno a 64 bit. Questo genere di cose possono però succedere anche con pdftex quando si esegue la divisione finale, se il divisore è troppo piccolo; In questo caso pdftex non può fare a meno di emettere un messaggio di errore, ma si può proseguire con le dita incrociate; i calcoli saranno sbagliati, quindi andrà corretto l'errore prima di ottenere risultati ragionevoli, ma lo si potrà fare senza perdere tutte le informazioni accessorie che pdftex è in grado di raccogliere o di generare durante la sua esecuzione.

Inconvenienti di questo genere possono succedere più sovente di quanto si immagini, quindi è opportuno fare sempre una analisi preventiva dei calcoli da eseguire, almeno come ordini di grandezza, per non ritrovarsi senza la possibilità di lasciare fare i conti a pdftex al fine di creare macro capaci di funzionare con qualunque insieme di valori degli argomenti.

[3]Non si badi al fatto che le indicazioni contengano solo le misure e non le unità di misura. È voluto per due motivi: (*a*) l'elaboratore usa solamente numeri, e (*b*) qui non si vuole annoiare il lettore con lunghe scritture di sequenze di bit o di cifre esadecimali od ottali; il concetto è quello di mostrare come si svolgono i conti, anche se qui usiamo le misure dei millimetri, non dimentichiamoci che i contatori numerici non possono contenere un numero equivalente di millimetri superiore a poco più di 5758.

29.8.4 Il pacchetti *ifthen* e *etoolbox*

Il pacchetti *ifthen* e *etoolbox* permettono di accedere ai comandi primitivi di controllo del flusso delle informazioni di cui TEX è capace. Il pacchetto *ifthen* fa parte di ogni distribuzione anche minimale del sistema TEX, mentre *etoolbox* non ne fa parte e, se la distribuzione non è completa o almeno quasi completa, bisogna installarlo apposta. Questo potrebbe essere un piccolo problema per gli utenti della distribuzione MiKTEX, che spesso dispongono di una distribuzione minimale o poco più. Alla fine del paragrafo si cercherà di illustrare un poco *etoolbox* che è una "scatola di attrezzi" molto completa e non gestisce solo i comandi condizionali, ma gestisce anche una miriade di altre funzionalità.

29.8.4.1 Il pacchetto *ifthen*

I comandi messi a disposizione da *ifthen* sono i seguenti.

`\ifthenelse` con la sintassi:

`\ifthenelse{`⟨*test*⟩`}{`⟨*esegui quando è vero*⟩`}{`⟨*esegui quando è falso*⟩`}`

esegue il ⟨*test*⟩ e se questo ⟨*test*⟩ restituisce il valore 'vero' allora vengono passati al flusso di informazioni da elaborare i token che formano il contenuto del primo argomento dopo il test, ⟨*esegui quando è vero*⟩. Altrimenti vengono eseguiti i token che formano il testo di ⟨*esegui quando è falso*⟩.

I ⟨*test*⟩ che si possono eseguire sono i seguenti.

confronto numerico con la sintassi

⟨$numero_1$⟩ *operatore* ⟨$numero_2$⟩

dove *operatore* è uguale a `>`, oppure `=`, oppure `<`; i numeri ⟨$numero_i$⟩ possono essere i contenuti di due contatori o di un contatore e di una quantità esplicita. Per i contatori LATEX bisogna specificarne il valore attraverso la macro `\value`.

`\equal` con la sintassi:

`\equal{`⟨$stringa_1$⟩`}{`⟨$stringa_2$⟩`}`

confronta due stringhe e se sono assolutamente identiche il test è vero, altrimenti è falso.

`\lengthtest` con la sintassi:

`\lengthtest{`⟨$lungh_1$ *operatore* $lungh_2$⟩`}`

confronta due lunghezze (generalmente almeno una delle due è contenuta in un registro-lunghezza) e restituisce il valore 'vero' se le due lunghezza stanno nella relazione implicata dall'*operatore*. Questo può essere un solo segno matematico fra `=`, `>`, oppure `<`.

`\isodd` con la sintassi:

`\isodd{`⟨*numero*⟩`}`

controlla se un numero (generalmente contenuto in un contatore) sia dispari.

`\isundefined` con la sintassi:

`\isundefined{`⟨`\comando`⟩`}`

restituisce il valore vero se ⟨`\comando`⟩ non è mai stato definito.

`\boolean` controlla lo stato di una variabile booleana; la sintassi per gestire queste variabili è la seguente.

`\newboolean{`⟨*variabile booleana*⟩`}`
`\provideboolean{`⟨*variabile booleana*⟩`}`
`\setboolean{`⟨*variabile booleana*⟩`}{`⟨*stato*⟩`}`
`\boolean{`⟨*variabile booleana*⟩`}`

dove ⟨*stato*⟩ è una delle due parole `true` (vero) oppure `false` (falso). In realtà `\boolean` è in grado di verificare lo stato anche delle variabili booleane interne a LaTeX, e anche di quelle definite con i comandi elementari di TeX o di Plain TeX. Con quest'ultimo si definisce un nuovo comando logico con `\newif` il quale accetta come argomento il comando per disteso e contemporaneamente definisce due altri comandi per impostare le corrispondenti variabili booleane. In pratica, se si volesse definire una nuova variabile booleana '`test`', con il pacchetto *`ifthen`* si dovrebbero usare i comandi

```
\newboolean{test}
...
\setboolean{test}{true}
...
\ifthenelse{\boolean{test}}{\textbf{Pippo}}{\textit{Pluto}}
```

Se invece si usassero i comandi di Plain TeX (accessibili anche quando si usa LaTeX) si dovrebbe scrivere

```
\newif\iftest
...
\testtrue
...
\iftest\textbf{Pippo}\else\textit{Pluto}\fi
```

Si è fatto questo esempio non tanto per invitare ad usare i comandi elementari di Plain TeX, quanto per permettere di capire come funzionano i test elementari che si trovano scritti a piene mani nei comandi definiti nei file di formato, di classe e di estensione.

Il comando `\provideboolean` esegue la stessa definizione che esegue `\newboolean`, ma omettendo ogni definizione se la ⟨*variabile booleana*⟩ esiste già.

`\and` `\or` e `\not` permettono di mettere insieme diverse frasi logiche da collegare fra di loro mediante questi operatori; l'intera frase comprendente gli operatori deve essere racchiusa fra `\(` e `\)`.

`\whiledo` consente di descrivere e realizzare un ciclo 'while'; la sintassi è:

`\whiledo{`⟨*test*⟩`}{`⟨*ciclo*⟩`}`

Questo comando ripete il ⟨*ciclo*⟩ fino a quando il ⟨*test*⟩, inizialmente vero, diventa falso. Va da sé che, prima di iniziare la ripetizione di ⟨*ciclo*⟩, gli elementi da cui dipende ⟨*test*⟩ devono essere inizializzati in modo tale che ⟨*test*⟩ sia vero. Il ⟨*ciclo*⟩ deve contenere delle istruzioni o dei comandi che prima o poi rendano il ⟨*test*⟩ falso, altrimenti LaTeX entra in un ciclo infinito e non ne esce più.

A titolo di esempio dell'uso di `\ifthenelse`, il comando `\cleardoublepage` è definito nel nucleo di LaTeX mediante la sintassi di basso livello TeX:

```
\def\cleardoublepage{\clearpage\if@twoside \ifodd\c@page\else
    \hbox{}\newpage\if@twocolumn\hbox{}\newpage\fi\fi\fi}
```

Usando il pacchetto *ifthen* questa definizione potrebbe essere tradotta in:

```
\newcommand*\cleardoublepage%
{\clearpage
  \ifthenelse{\(\boolean{@twoside}\and\not\isodd{\value{page}}\)}{%
      \mbox{}\newpage
      \ifthenelse{\boolean{@twocolumn}}{\mbox{}\newpage}{}%
                                  }{}%
}
```

La scrittura è un poco più complessa, ma si vedono meglio gli effetti e la natura dei test; il primo controlla se stiamo componendo fronte e retro e se siamo su una pagina pari; è chiaro che se non stiamo componendo fronte e retro, saltare una pagina per ricominciare da una pagina dispari, una pagina di destra, non ha molto senso; inoltre è chiaro che se la pagina è dispari non bisogna fare nulla. Infine controlla se si sta componendo su due colonne: in questo caso esegue comunque un `\newpage` che, a due colonne, vuol dire di interrompere una colonna e ricominciare a comporre nella colonna successiva[4]. L'incolonnamento dei contenuti delle varie clausole e delle relative parentesi graffe permette di seguire meglio i successivi passi.

[4] Il secondo `\ifthenelse` può essere omesso se il comando precedente, invece di `\newpage`, fosse nuovamente `\clearpage`.

29.8.4.2 Il pacchetto *etoolbox*

Il pacchetto *etoolbox* dispone di una miriade di nuovi comandi che certamente non verranno descritti qui di seguito, tranne alcuni che abbiano a che fare con i comandi condizionali. Queso pacchetto dispone di comandi per svolgere azioni importanti nei seguenti settori:

Definizioni di comandi estende le definizioni standard di LaTeX provvedendo anche a definizioni espanse e globali; provvede a definire comandi robusti e a "irrobustire" comandi fragili preesistenti.

Riparazioni o modifiche di codice provvede ad estendere o comunque a modificare il testo sostitutivo di macro esistenti.

Protezione di comandi Estende i meccanismi di protezione di LaTeX.

Gestione di lunghezze e contatori non solo estende la gestione delle lunghezze e dei contatori, ma permette di definire un numero maggiore di registri e di contatori ad un numero talmente alto che in pratica si può considerare illimitato. Toglie la barriera dei 256 contatori e registri di ogni tipo che possono essere usati con i programmi del sistema TeX.

Agganci ai comandi di gestione del documento LaTeX dispone di pochi comandi, come per esempio `\AtBeginDocument`, che possano differire l'esecuzione di certi comandi a posizioni speciali della gestione dei file sorgente. *etoolbox* estende enormemente queste capacità.

Agganci agli ambienti in modo analogo vengono definiti altri "ganci" (in inglese *hooks*) che permettono di gestire correttamente le istruzioni da eseguire al momento opportuno dentro ad un ambiente o "ai suoi estremi".

Comandi ad uso degli autori Mentre i comandi precedenti sono di interesse specialmente per chi scrive file di classe o di estensione, i prossimi comandi sono a disposizione degli autori.

Definizioni vengono messi a disposizione ulteriori comandi per gestire le definizioni di comandi.

Controllo dello sviluppo dei comandi anche questi ulteriori comandi servono per controllare al tempistica dello sviluppo dei comandi.

Gestione dei *ganci* anche questi comandi permettono di gestire i "ganci attaccati ad altri comandi.

Modifica di comandi vale quanto detto spora per gli strumenti messi a disposizione dei programmatori di classi e di estensioni.

Variabili booleane è questa la parte che ci interessa di più in questo paragrafo e ce ne occuperemo fra un poco.

Test in generale estende i comandi per eseguire test senza fare uso esplicitemante di variabili booleane.

Elaborazione di liste questo è un argomento poco coperto dal linguaggio di LaTeX e gli strumenti di questa sezione erano attesi da tempo.

Miscellanea di strumenti vari vengono aggiunti ancora alcuni strumenti non classificabili nelle categorie precedenti; servono principalmente per lavorare con numeri romani come se fossero numeri in cifre arabe.

La documentazione è ben fatta e succinta; qualunque utente che voglia cimentarsi con la scrittura di macro farebbe bene non solo ad adottare l'uso sistematico di *etoolbox*, ma anche a studiarsi con attenzione la descrizione dei singoli comandi dai quali si può imparare molto.

Il comando `\newbool` non è molto diverso dal comando analogo del pacchetto *ifthen*; la differenza sta nel fatto che esegue una verifica che il nome sia disponibile per una definizione; il comando associato `\providebool` provvede alla definizione solo se la variabile non è ancora definita; Come con il pacchetto *ifthen* sono da considerasi variabili booleane i nomi formati dalle stringhe di caratteri che seguono `\if` in tutti i comandi condizionali definiti con `\newif` e lo sono anche le stringe che seguono la stringa `\if` dei comari primitivi: per esempio, la variabile booleana `odd` legata al test primitivo `\ifodd`; la variabile booleana `mmode` legata al test primitivo `\ifmmode`.

Questo pacchetto *etoolbox* però esegue i test sul valore della variabile booleana mediante un comando più diretto:

`\ifbool{`⟨*variabile*⟩`}{`⟨*esegui se vero*⟩`}{`⟨*esegui se falso*⟩`}`

Nello stesso modo ci sono comandi più semplici per impostare una variabile a vero o falso, o per impostarla come si farebbe con *ifthen*. Interessante è anche il comando negato:

`\notbool{`⟨*variabile*⟩`}{`⟨*esegui se non vero*⟩`}{`⟨*esegui se non falso*⟩`}`

Utilissimi sono i comandi per verificare se un comando esista già (`\ifdef`) o anche se il nome di un comando corrisponda a un comando esistente (`\ifcsdef`). Bisogna ricordarsi che il nome di un comando è il comando senza il backslash iniziale; ma quando si specifica il nome di un comando lo si può fare anche senza rispettare le regole che sia formato solo da lettere, oppure solo da un unico carattere non alfabetico. Così esistono `\ifundef` e `\ifcsundef`. Ma esistono anche `\ifdefprefix` e `\ifcsprefix` per sapere se una macro è stata definita con il prefisso `\long` o il prefisso `\protect`; una macro di tipo "long" accetta come argomenti uno o più capoversi, mentre se non ha ricevuto al momento della sua definizione il prefisso `\long` può accettare come argomento solo una stringa che non contenga né direttamente né indirettamente nemmeno una fine di paragrafo. `\protect` caratterizza le macro presumibilmente robuste.

Molto interessanti i comandi `\ifdefempty` e `\ifcsempty` per sapere se il testo di sostituzione di una macro è vuoto; `\ifdefequal` e `\ifcsequal` per sapere se due macro hanno lo stesso numero di argomenti e lo stesso testo sostitutivo oltre

agli stessi prefissi; `\ifdefstring` e `\ifcsstring` per sapere se il testo sostitutivo di una macro è uguale ad una stringa di caratteri data.

Comandi analoghi esistono per testare il nome dei contatori TeX (definiti con `\newcount`) o LaTeX(definiti con `\newcounter`), e di lunghezze

Poi ci sono test per confrontare stringhe di caratteri indipendenti dalla loro categoria di appartenenza; questi confronti implicano che le due stringhe non siano esplicite entrambe; per esempio si potrebbero usare dentro la definizione di una macro per controllare se un parametro sia vuoto oppure sia costituito da uno spazio; per esempio:

```
\newcommand\prova[1]{%
\iffblank{#1}{esegui se vero}{esegui se falso}}
```

controlla se l'unico parametro passato alla macro quando viene chiamato sia vuoto o sia uno spazio, poi si comporta conseguentemente.

Il confronto fra numeri è gestito con `\ifnumcomp` con la sintassi:

`\ifnumcomp`⟨*test numerico*⟩`{`⟨*esegui se vero*⟩`}{`⟨*esegui se falso*⟩`}`
dove:
⟨*test numerico*⟩
è:
`{`⟨*prima espressione numerica*⟩`}{`⟨*relazione*⟩`}{`⟨*seconda espressione numerica*⟩`}`

Si noti che la relazione è espressa mediante i soliti segni `=`, `>` e `<`. Invece è interessante che ciacuna delle due ⟨*espressioni numeriche*⟩ sia qualunque espressione numerica che abbia senso fra i delimitatori `\numexpr` e `\relax`, senza però contenere questi delimitatori; ci si ricordi che le espressioni numeriche e di lunghezze vengono arrotondate e non troncati per cui un test del tipo:

`\ifnumcomp{6/2}{=}{5/2}{`⟨*esegui se vero*⟩`}{`⟨*esegui se falso*⟩`}`

procede ad eseguire ⟨*esegui se vero*⟩ perché $6/2 = 3$ (e su questo non ci sono dubbi) ma $5/2 = 2{,}5$ per cui le regole dell'arrotondamento portano il risultato finale a 3.

Il comando generale `\ifnumcomp` si semplifica e diventa più immediato da capire se invece di specificare il segno di relazione si usa uno dei tre comandi `\ifnumequal`, `\ifnumgreater` e `\ifnumless`; il comando primitivo `\ifodd` diventa `\ifnumodd` ma riceve i soliti due argomenti per eseguire cose diverse a seconda che il test sia vero o falso.

Analoghi comandi servono per controllare due espressioni dimensionali.

I comandi per le espressioni logiche sono superficialmente simili nel senso che per singoli valori non sono necessarie le parentesi tonde, ma se queste ci sono servono per alterare l'ordine di esecuzione delle operazioni logiche; in più *etoolbox* consente di gestire ulteriori operatori logici oltre ai soliti `and`, `or` e `not`, si noti però che ora gli operatori *non* sono comandi, ma semplici parole.

Il vantaggio nell'usare le macro di *etoolbox* è che sono molo più numerosi e svolgono con semplicità numerose funzioni che richiederebbero scritture più

lunghe rispetto a `\ifthen`, ma per lo più si tratta di comandi robusti e quindi possono venire usati senza paura che possano dare luogo a errori di difficile comprensione.

29.9 Spaziature

Gli spazi orizzontali e verticali possono venire inseriti a mano con i seguenti comandi:

```
\hspace{⟨lunghezza⟩}
\hspace*{⟨lunghezza⟩}
\vspace{⟨lunghezza⟩}
\vspace*{⟨lunghezza⟩}
```

I comandi con asterisco impediscono che gli spazi siano eliminati alla fine o all'inizio di una riga o all'inizio o alla fine di una pagina, come invece avviene con gli spazi 'semplici'. Questa possibilità di essere eliminati è essenziale per la buona composizione di una pagina e delle sue righe, ma può non essere quello che si desidera quando si compone dentro ad una scatola oppure quando si eseguono composizioni speciali: si pensi per esempio ad un frontespizio dove non si vuole inserire il primo elemento scritto all'inizio della gabbia, ma lo si desidera spostare in basso per l'equilibrio della pagina.

Molto comodi risultano i comandi abbreviati

```
\bigskip
\medskip
\smallskip
```

che equivalgono rispettivamente a

```
\vspace{\bigskipamount}        \vspace{\medskipamount}
\vspace{\smallskipamount}
```

Le lunghezze indicate con i tre comandi `\...skipamount` sono definiti nei file di classe, ma generalmente essi hanno i valori di 12 pt, 6 pt e 3 pt.

Il comando

```
\addvspace{⟨lunghezza⟩}
```

aggiunge spazio verticale tenendo conto di eventuale spazio verticale già inserito implicitamente da comandi precedenti, così che alla fine lo spazio complessivo aggiunto ammonti esattamente alla ⟨*lunghezza*⟩ specificata.

I comandi `\hfill` e `\vfill` equivalgono rispettivamente a `\hspace{\fill}` e `\vspace{\fill}`.

29.10 Figure, tabelle ed altri oggetti flottanti

I comandi per rendere flottanti le figure e le tabelle sono descritti nei prossimi paragrafi.

29.10.1 Figure e tabelle

```
\begin{figure}[⟨posizione⟩] ⟨figura⟩ \end{figure}
\begin{figure*}[⟨posizione⟩] ⟨figura⟩ \end{figure*}
\begin{table}[⟨posizione⟩] ⟨tabella⟩ \end{table}
\begin{table*}[⟨posizione⟩] ⟨tabella⟩ \end{table*}
```

I comandi senza asterisco vanno bene sia componendo ad una colonna, sia componendo a due colonne. I comandi con l'asterisco inseriscono l'oggetto a larghezza piena in testa ad una pagina composta a due colonne; ma componendo ad una colonna si può usare indifferentemente la versione con o quella senza asterisco.

Le posizioni possibili sono

`t`	in testa alla pagina
`b`	al piede della pagina
`h`	'qui', se possibile...
`p`	in una pagina di soli oggetti flottanti

Con tutte le classi standard, se l'argomento facoltativo ⟨*posizione*⟩ non è specificato, le posizioni di default sono `tbp`.

La lista delle posizioni in realtà accetta anche `!` che invita LaTeX a 'mettercela tutta' per collocare l'oggetto anche trascurando alcuni dei parametri di posizionamento dettati dalla classe in uso.

LaTeX può mettere un filetto sotto a una figura in testa alla pagina, oppure sopra una figura in calce alla pagina; può mettere anche un filetto sotto ad una figura a piena pagina in testa ad una pagina composta a due colonne. I comandi per mettere questi filetti sono `\topfigrule`, `\botfigrule` e `\dblfigrule`; di default questi tre comandi sono equivalenti a `\relax`, quindi non fanno niente, Ogni classe oppure ogni compositore li può ridefinire in modo che mettano qualcosa, generalmente un filetto, ma potrebbe essere una qualunque decorazione, nella posizione a cui si riferiscono, ma questo qualcosa deve avere altezza *nulla* allo stesso modo del filetto che separa le note in calce dal testo sovrastante. La definizione di questo filetto `\footnoterule` è la seguente:

```
\def\footnoterule{\kern-3\p@ \hrule \@width 2in \kern 2.6\p@}
```

e basta definire i comandi di questi filetti cambiando il nome `\footnoterule` con `\topfigrule`, o `\botfigrule`, o `\dblfigrule`; per esempio:

```
\def\topfigrule{\kern2.6\p@ \hrule \@width 2in \kern 3\p@}
\def\botfigrule{\kern-3\p@ \hrule \@width 2in \kern 2.6\p@}
\def\dblfigrule{\kern-2.6\p@ \hrule \@width 2in \kern 3\p@}
```

Si noti l'espediente per fare sembrare il filetto di altezza nulla; si arretra di 3pt, si mette un filetto spesso 0,4pt, e si avanza di 2,6pt; complessivamente non si è avanzato in verticale di nessuno spazio. gli spostamenti in avanti o all'indietro di 2,6pt e di 3pt sono messi strategicamente in modo che lo spostamento positivo separi il filetto da ciò che esso segue o precede.

LaTeX, durante la costruzione della pagina di uscita, segue dei criteri di ottimalità per collocare gli oggetti flottanti che eventualmente sono accodati per essere emessi appena possibile. Questi criteri implicano il rispetto di certe frazioni minime o massime di spazio dedicato a questa o quella parte del documento, nonché al numero massimo di oggetti che possono essere collocati in una pagina. Le regole che LaTeX segue sono descritte qui di seguito, ma è chiaro che se si vuole che LaTeX faccia diversamente da come è stato programmato, bisogna comprendere appieno queste regole e bisogna capire come funzionano i parametri massimi o minimi che vi presiedono.

1. LaTeX colloca un oggetto flottante nel primo posto che non viola le regole seguenti, salvo che, se è specificato il codice `h`, questo ha la precedenza sul codice `t`.

2. LaTeX non collocherà mai un oggetto flottante in una pagina che viene prima di quella dove compare il testo del file sorgente che attornia l'ambiente di flottaggio.

3. LaTeX colloca le figure in ordine e lo stesso fa con le tabelle, per cui non può succedere che la figura 22 appaia prima della figura 21.

   Va notato però che quando si compone a due colonne (senza usare il pacchetto *multicol*[5], ma usando la specifica LaTeX `\twocolumns`) possono formarsi pagine che hanno figure o tabelle a giustezza piena contemporaneamente a una o più figure o tabelle con la giustezza di una colonna. In questi casi, visto che gli oggetti a piena giustezza vengono collocate solo in testa, mentre gli oggetti con la giustezza di una colonna possono essere collocati sia in testa alla colonna, sia in calce, sia in una posizione qualsiasi consentita dai parametri di posizione, può succedere che l'oggetto in testa abbia un numero successivo a quello di un oggetto in colonna. Per ovviare a questo inconveniente è necessario usare un piccolo pacchetto di correzioni del nucleo di LaTeX, *fixltx2e*, che esegue alcune modifiche alla routine di emissione della pagina composta, onde evitare questa e poche altre apparenti incongruenze presenti nel nucleo standard. La documentazione di questo pacchetto si trova in `.../doc/latex/base/`, ma l'utente generalmente deve usare questo pacchetto solo se compone a due

[5] Questo pacchetto non consente di mettere figure all'interno delle colonne.

colonne e se ha numerose figure o tabelle di varie dimensioni orizzontali, altrimenti questi difettucci non si manifestano quando si compone ad una sola colonna.

4. LaTeX colloca gli oggetti flottanti solamente nelle posizioni specificate nell'argomento facoltativo ⟨*posizione*⟩ o, in mancanza di questo, in una delle tre posizioni di default `tbp`.

5. LaTeX non collocherà mai un oggetto flottante in una pagina che non contiene abbastanza spazio, quindi non produrrà mai una '`Overfull vbox`' a causa degli oggetti flottanti. Be', a meno che un oggetto flottante di per sé non sia così grande da eccedere lo spazio disponibile; a questo il compositore deve provvedere preventivamente mediante le opportune operazioni di scalamento e gli altri artifici descritti negli appositi capitoli relativi a figure e tabelle.

6. I vincoli imposti dai parametri stilistici tipici di ogni classe non vengono mai violati; quando viene specificato `!` vengono rilassati i vincoli imposti dai parametri che contengono sia testo sia oggetti flottanti, mentre LaTeX continua a rispettare i vincoli imposti alle pagine che contengono solamente oggetti flottanti. Per queste pagine di soli oggetti flottanti le regole vengono ignorate quando si emettono i comandi `\clearpage` oppure `\cleardoublepage`, e, ovviamente, quando si incontra `\end{document}`, perché questi comandi e la specifica che il documento è terminato ordinano a LaTeX di svuotare le code di tutto ciò che si trova ancora in memoria.

Quando si specifica la ⟨*posizione*⟩ bisogna dare abbastanza possibilità a LaTeX di fare il suo mestiere, altrimenti l'oggetto che non ha altre possibilità che una sola, blocca le code fino alla fine del documento o alla prima esecuzione di `\clearpage` o `\cleardoublepage`; questi comandi vengono emessi automaticamente quando si inizia un nuovo capitolo, ma, a parte che è brutto vedere tutte o quasi le figure di un capitolo accumulate alla sua fine, bisogna ricordare che la memoria di LaTeX è programmata per memorizzare solamente 18 oggetti flottanti; se dovessero accodarsene di più, quelli in eccesso verrebbero persi e verrebbe emesso un messaggio che avvisa del fatto, ma è una magra consolazione. Si esamini anche la possibilità di usare il pacchetto *morefloats*.

I comandi che si possono usare dentro agli ambienti di flottaggio tipicamente sono:

`\caption` con la sintassi

`\caption[`⟨*didascalia breve*⟩`]{`⟨*didascalia*⟩`}`

dove se non si specifica la ⟨*didascalia breve*⟩ questa viene automaticamente resa identica alla didascalia 'lunga'; potrebbe non essere una buona idea quella di inviare alla lista delle figure o delle tabelle l'intera didascalia,

specialmente se questa contiene uno o più periodi dopo il primo che svolge il compito di titolo, mentre i periodi successivi svolgono il compito di fornire maggiori delucidazioni sull'oggetto specifico.

`\label` può essere usato, anzi è conveniente che sia usato per ogni figura e per ogni tabella; il comando deve essere collocato dopo il comando `\caption`, perché finché questo non viene eseguito, all'oggetto flottante non è ancora stato assegnato un numero.

`\suppressfloats` con la sintassi

```
\suppressfloats[⟨posizione⟩]
```

può essere collocato fuori degli ambienti di flottaggio al fine di evitare che LaTeX metta altri oggetti nella stessa ⟨*posizione*⟩ specificata come argomento facoltativo; se non si specifica nulla, tutti gli oggetti flottanti sono esclusi dalla pagina corrente; Se però nella ⟨*posizione*⟩ del comando di apertura degli ambienti *figure* o *table* compare `!`, allora `\suppressfloats` viene ignorato.

I parametri dimensionali e numerici che regolano il deflusso degli oggetti flottanti dalle rispettive code sono i seguenti.

`topnumber` è il nome del contatore che contiene il numero *massimo* di oggetti flottanti che possono essere collocati in testa alla pagina.

`\topfraction` è la *massima* frazione di pagina destinata agli oggetti flottanti in testa ad una pagina, se questa contiene anche del testo.

`bottomnumber` è il contatore che contiene il numero *massimo* di oggetti flottanti in calce alla pagina.

`\bottomfraction` è la *massima* frazione di pagina destinata agli oggetti flottanti in calce ad una pagina, se questa contiene anche del testo.

`totalnumber` è il contatore che contiene il *massimo* numero di oggetti flottanti che possono comparire in una pagina di testo, indipendentemente dal fatto che che siano in testa o in calce o in mezzo al testo.

`\textfraction` per una pagina che contenga anche del testo, è la frazione *minima* della pagina destinata al testo.

`\floatpagefraction` in una pagina di soli oggetti flottanti rappresenta la *minima* frazione di pagina che deve essere occupata da questi oggetti; se cioè si specifica il parametro di posizione `p` per un oggetto flottante e questo è troppo piccolo, esso viene trattenuto in memoria finché non si trova un altro oggetto flottante dello stesso genere e con lo stesso attributo di posizione che assieme possano superare questo valore minimo specificato.

dbltopnumber il nome del contatore che contiene il numero *massimo* di oggetti flottanti a piena pagina da mettere in testa ad una pagina con il testo composto su due colonne.

\dbltopfraction la *massima* frazione di pagina a giustezza piena da destinare agli oggetti flottanti in testa alla pagina quando si compone a due colonne.

\dblfloatpagefraction è la *minima* frazione di pagina da occupare con oggetti flottanti a giustezza piena quando si compone una pagina di soli oggetti flottanti in un testo composto a due colonne. È, insomma, l'analogo di **\floatpagefraction** che, invece, vale quando si compone il testo ad una sola colonna.

\floatsep è la distanza *minima* da riempire con un contrografismo incolore fra due oggetti flottanti consecutivi.

\textfloatsep è la distanza *minima* da interporre fra gli oggetti flottanti in testa o in calce e il testo adiacente.

\intextsep è la distanza *minima* fra un oggetto flottante a centro pagina e il testo che lo precede e il testo che lo segue.

\dblfloatsep è la distanza *minima* fra due oggetti flottanti consecutivi in testa ad una pagina composta a due colonne.

\dbltextfloatsep è la distanza *minima* posta fra gli oggetti flottanti in testa ad una pagina composta a due colonne e il testo sottostante.

I contatori si impostano con i comandi specifici, in particolare **\setcounter**; le frazioni **\...fraction** sono valori decimali, in generale fratti e minori dell'unità che vengono conservati dentro le macro con i rispettivi nomi, e quindi tutti si reimpostano, per modificare i valori di default, mediante **\renewcommand**. I separatori **\...sep** sono lunghezze e si impostano con il comando **\setlength**. Ogni classe definisce i suoi particolari valori per questi parametri; per la classe *book* composta in corpo 10 (come questo testo) i parametri sono riportati nella tabella 29.3.

Va detto che i 'Numeri' e le 'Frazioni' indicati nelle tabelle 29.3 non devono necessariamente essere coerenti; sembra strano, ma è comprensibile. Per i 'Numeri', per esempio, **totalnumber** sembra essere la somma di **topnumber** e **bottomnumber**; se però **totalnumber** fosse posto al valore 2, questo vorrebbe dire che *al massimo* ci possono essere 2 oggetti flottanti in testa e uno in calce, ma in ogni caso non possono essere tutti e tre presenti, perché nella pagina ce ne possono essere *al massimo* due. Lo stesso vale per le frazioni; esse rappresentano dei limiti per delle disuguaglianze; questi limiti non debbono necessariamente avere per somma l'unità; l'importante è che queste disuguaglianze siano tutte rispettate nell'ordine di preferenza che LaTeX assegna alla collocazione degli oggetti flottanti.

Numeri		Frazioni	
`topnumber`	2	`\topfraction`	0,7
`bottomnumber`	1	`\bottomfraction`	0,3
`totalnumber`	3	`\textfraction`	0,2
`dbltopnumber`	2	`\floatpagefraction`	0,5
		`\dbltopfraction`	0,7
		`\dblfloatpagefraction`	0,5

Separatori	
`\floatsep`	`12pt plus 2pt minus 2pt`
`\textfloatsep`	`20pt plus 2pt minus 4pt`
`\intextsep`	`12pt plus 2pt minus 2pt`
`\dblfloatsep`	`12pt plus 2pt minus 2pt`
`\dbltextfloatsep`	`20pt plus 2pt minus 4pt`

Tabella 29.3: Parametri nella classe *book* composta in corpo 10 per gestire gli oggetti flottanti

Il compositore si ricordi, specialmente per quel che riguarda le frazioni, che nel computo dell'ingombro dell'oggetto flottante bisogna tenere conto anche dello spazio destinato alla didascalia e del suo spazio di separazione. Spesso il compositore è deluso dalla rigidezza con cui LaTeX gestisce gli oggetti flottanti; in realtà LaTeX, come tutti i programmi, si comporta come gli è stato prescritto, in particolare, in questo caso, come richiesto dai 'Numeri', dalle 'Frazioni' e dai 'Separatori' che compaiono nella tabella 29.3. Se non piacciono quei valori, li si può cambiare, ma il compositore stia attento che la cura non sia peggio del male. Piuttosto egli dedichi la sua attenzione al dimensionamento degli oggetti flottanti, all'eliminazione di spazi inutili al loro contorno, all'eliminazione delle parti inutili delle fotografie che si trovano spessissimo ai margini laterali o verticali; insomma usi appropriatamente le possibilità offerte dalle opzioni di `\includegraphics` perché è con quelle che si curano davvero sia il deflusso degli oggetti flottanti dalle rispettive code, sia la qualità del documento prodotto.

29.10.2 Note marginali

I comandi che portano alla composizione e alla collocazione delle note marginali sono i seguenti:

```
\marginpar[⟨nota di sinistra del testo⟩]{⟨nota di destra del testo⟩}
\reversemarginpar
\normalmarginpar
```

La posizione normale delle note quando si compone ad una colonna è il margine esterno; quando si compone a due colonne è il margine adiacente alla colonna alla

quale la nota si riferisce, quindi il margine di sinistra per la colonna di sinistra e il margine di destra per la colonna di destra; e questo avviene indipendentemente dal fatto che ci si trovi in una pagina di sinistra o di destra. `\reversemarginpar` scambia i margini per le note composte ai margini di testi composti ad una sola colonna e `\normalmarginpar` ripristina la collocazione di default. Questi due comandi sono inattivi quando si compone a due colonne.

Per il testo delle note che dovrebbero apparire a sinistra del testo a cui si riferiscono potrebbe essere applicata la dichiarazione `\raggedleft` per avere una composizione in bandiera giustificata a destra. Tuttavia questo è consigliabile solo quando le annotazioni marginali sono sempre molto brevi in modo da non superare una riga. Cambiare giustificazione da nota a nota sarebbe una caduta di uniformità e di stile. D'altra parte le note di più righe giustificate solo a destra sono leggermente più laboriose da leggere rispetto alle note giustificate solo a sinistra; il compositore ci pensi bene prima di decidere come comporre le note; eventualmente si crei una macro che componga sempre entrambi i testi di sinistra e di destra uniformemente con lo stesso stile, in modo da non dover operare a mano così da evitare i possibili errori di disomogeneità.

I parametri che governano la collocazione delle note, a parte il margine 'normale' o 'scambiato', sono:

`\marginparwidth` è la giustezza delle note marginali e deve essere evidentemente minore della larghezza del margine fisico a lato del testo.

`\marginparsep` è lo spazio di separazione fra il testo e il blocchetto della nota marginale.

`\marginparpush` è lo spazio con cui una nota marginale viene spostata in basso rispetto alla fine della nota marginale precedente; talvolta questo obbliga a portare la nota nella pagina successiva, ma viene emesso un avvertimento sullo schermo e nel file `.log`. Per riparare queste situazioni non tanto desiderabili, bisogna necessariamente riformulare il testo principale a cui la nota in questione si riferisce, oppure, bisogna accorciare il testo della nota precedente.

Un avvertimento: LaTeX tratta le note marginali alla stessa stregua degli oggetti flottanti; questo vuol dire che usa le stesse scatole per conservare in memoria gli oggetti flottanti; siccome le scatole a disposizione sono solo 18, l'uso di frequenti note marginali e di molte figure e/o tabelle aumenta il rischio di esaurire la memoria a disposizione e di perdere degli oggetti flottanti.

Si può fare uso del pacchetto *morefloats* che raddoppia il numero di oggetti flottanti in coda; permette di arrivare fino a 36. Il motivo per il quale esso arriva solo fino a una coda di 36 oggetti flottanti è per evitare di esaurire la *main memory* del programma di composizione. Tuttavia va osservato che il pacchetto fu scritto nel 1990, quindi quando la memoria a disposizione dei vari main frame e dei PC personali era veramente piccola. Il sistema TeX che sto usando dispone di una memoria di 3 000 000 parole di 4 byte per svolgere il suo lavoro, ma ne usa solamente la settima parte, come si può leggere alla fine del file `.log`.

29.11 Incolonnamenti

29.11.1 L'ambiente *tabbing*

Nel testo non se ne è nemmeno parlato, perché si ritiene che questo ambiente sia una reminiscenza degli incolonnamenti eseguiti in dattilografia; quando TEX 78 nacque, le macchine da scrivere meccaniche ed elettriche erano ancora diffusissime e le tabulazioni eseguite con il tasto di tabulazione, nonché con gli arresti di tabulazione, erano familiari a tutti. Tuttavia l'ambiente *tabbing*, nato, forse, proprio per affidarsi a qualcosa di noto a qualunque dattilografo, presenta dei notevoli inconvenienti, primo fra i quali quello di essere un ambiente fragile e che non può essere annidato dentro altri ambienti.

La sintassi è quella comune a qualunque ambiente:

`\begin{tabbing}` ⟨*testo da tabulare*⟩ `\end{tabbing}`

I comandi per gestire gli arresti di tabulazione sono i seguenti.

\= serve per impostare un arresto di tabulazione; spesso conviene scrivere una riga modello da *non* comporre se si ricorre al comando seguente.

\kill non compone la riga che precede questa parola chiave, ma conserva le informazioni di impostazione degli arresti di tabulazione che la riga conteneva.

\> serve per spostare la composizione al successivo arresto di tabulazione.

**** comincia una nuova riga riportando il contatore degli arresti di tabulazione al valore iniziale, generalmente zero.

\+ incrementa di una unità il valore iniziale del contatore di tabulazione così che da questo momento in poi e fino ad ordine contrario, tutte le righe risultano rientrate e incolonnate sotto un arresto presente nella riga precedente.

\- decrementa di una unità il valore iniziale del contatore di tabulazione; contrasta l'effetto di **\+**.

\< torna indietro di un arresto di tabulazione e può essere usato solamente all'inizio di una riga.

\' serve per spostare tutto il contenuto di una 'colonna' a filo del margine destro del suo campo di tabulazione.

\` serve per comporre il contenuto della sua colonna a filo del margine sinistro del suo arresto di tabulazione.

\pushtabs manda in memoria le posizioni degli attuali arresti di tabulazione, consentendo di impostarne di diversi.

\poptabs serve per richiamare dalla memoria un insieme di arresti di tabulazione precedentemente memorizzati.

\a= `\a'` e `\a‘` agiscono come `\=`, `\'` e `\‘` per comporre gli accenti macron (lungo), acuto e grave rispettivamente, perché questi comandi, come si vede sopra, sono stati ridefiniti all'interno dell'ambiente per svolgere funzioni diverse. Il compositore che usa il pacchetto *inputenc* non se ne preoccupi e continui ad usare tranquillamente le lettere accentate della sua tastiera, perché quel pacchetto provvede alla compatibilità senza che il compositore debba preoccuparsene più di tanto. Il compositore deve purtroppo occuparsene se e solo se la sua tastiera è priva di segni accentati e se il suo *shell editor* non consente di usare combinazioni di tasti per inserire direttamente nel file `.tex` i segni accentati, oppure, infine, se ha impostato il suo *shell editor* in modo che salvi i file in modo tale che contengano solo caratteri ASCII a 7 bit, così da assicurare la massima portabilità 'cross-platform'.

29.11.2 Gli ambienti *array* e *tabular*

Le sintassi sono:

```
\begin{array}[⟨allineamento⟩]{⟨colonne⟩} ⟨righe⟩ \end{array}
\begin{tabular}[⟨allineamento⟩]{⟨colonne⟩} ⟨righe⟩ \end{tabular}
\begin{tabular*}{⟨larghezza⟩}[⟨allineamento⟩]{⟨colonne⟩}
    ⟨righe⟩
\end{tabular*}
```

La ⟨*posizione*⟩ serve per stabilire l'allineamento della tabella o della matrice con il 'testo' circostante; la ⟨*larghezza*⟩ serve per imporre una larghezza specificata alla tabella. I descrittori delle ⟨*colonne*⟩ sono certi codici che ora si rivedranno; le ⟨*righe*⟩ sono i contenuti delle celle orizzontali contenenti il testo o la matematica da comporre. *array* si usa solo in ambiente matematico; gli ambienti *tabular* e *tabular** solo in modo testo; sia gli uni sia gli altri ambienti possono contenere singole celle contenenti 'testo' dell'altra specie, ma bisogna specificarlo per ciascuna di queste celle; in ambiente *tabular* basta usare gli appositi comandi per passare alla matematica; in ambiente *array* si possono usare i segni di dollaro che agiscono da interruttori/deviatori; se si è già in modo matematico (come avviene per ogni cella di un *array*) il primo segno di dollaro passa in modo testo e il secondo riporta al modo matematico.

I codici di allineamento sono:

`t` serve per allineare la riga di testa con il testo circostante.

`b` serve per allineare la riga di base con quella del testo circostante.

`c` non necessita di essere espresso, perché l'allineamento centrato è quello di default.

I descrittori delle ⟨*colonne*⟩ e i loro separatori sono i seguenti.

`l` colonna con i contenuti allineati a sinistra.

`c` colonna con i contenuti centrati.

`r` colonna con i contenuti allineati a destra.

`|` filetto verticale di separazione fra colonne adiacenti.

`\vline` lo stesso filetto quando deve essere inserito in una *@-espressione*.

`@{`⟨*testo*⟩`}` *@-espressione*; si tratta di un particolare costrutto con il quale si specifica che ⟨*testo*⟩ sostituisce completamente il separatore ordinario fra due colonne adiacenti. Questo significa che se i contenuti delle due celle adiacenti devono essere in qualche modo distanziati, allora gli spaziatori devono essere inseriti dentro la *@-espressione*. Un elemento importante delle *@-espressioni* sono le dichiarazioni di spaziatura con elasticità 'infinita' da usare con l'ambiente *tabular** per consentire che queste particolari tabelle si possano estendere fino a raggiungere la larghezza specificata.

`\extracolsep{`⟨*larghezza*⟩`}` specificato dentro una *@-espressione* serve a dichiarare che lo spaziatore con la ⟨*larghezza*⟩ esplicitata venga inserito a sinistra del contenuto di tutte le celle *che seguiranno* sulla stessa riga; quindi *non* di questa e di tutte le celle che seguiranno, ma solo di quelle che seguiranno. Tipicamente questo comando viene inserito nella prima *@-espressione* che compare a sinistra della prima cella di una riga; quindi questa cella non verrà mai allargata per raggiungere la giustezza desiderata.

`\fill` è una larghezza naturalmente nulla ma dotata di allungamento infinito, tipicamente usata come argomento di `\extracolsep`.

`p{`⟨*larghezza*⟩`}` serve per descrivere una colonna composta come un (breve) capoverso la cui prima riga è allineata con quella delle celle adiacenti e avente una giustezza pari a ⟨*larghezza*⟩. Il testo delle colonne composte con questo descrittore non può contenere direttamente il comando `\\` perché LaTeX non potrebbe sapere se esso si riferisce alla cella o all'intera riga di celle. Perciò è necessario racchiudere questo comando dentro un ambiente, come *minipage* o un altro *tabular*, oppure dentro una scatola `\parbox` esplicita; oppure dentro il raggio di azione di comandi come `\centering`, `\raggedright` oppure `\raggedleft`, purché a loro volta siano contenuti dentro un gruppo delimitato da parentesi graffe oppure siano contenuti all'interno di altri ambienti.

`*{`⟨*numero*⟩`}{`⟨*descrittori*⟩`}` è una forma abbreviata per ripetere ⟨*numero*⟩ volte le stessa sequenza di ⟨*descrittori*⟩ delle colonne. Una `*`-*espressione* ne può contenere un'altra; si consiglia di non eccedere con queste forme stenografiche innestate l'una nell'altra altrimenti dopo un po' non si capisce più che cosa si sia specificato. Piuttosto è meglio rifarsi alle possibilità offerte dal pacchetto *array* e dal comando `\newcolumntype`.

I comandi che si possono usare dentro gli ambienti di incolonnamento sono i seguenti.

`\multicolumn` con la sintassi

> `\multicolumn{`⟨*numero*⟩`}{`⟨*descrittore*⟩`}{`⟨*cella*⟩`}`
>
> serve per comporre un'unica ⟨*cella*⟩ con il ⟨*descrittore*⟩ specificato che occupi ⟨*numero*⟩ celle adiacenti. Il ⟨*descrittore*⟩ oltre alla collocazione con una delle varie lettere chiave (compresa la `p`) può contenere filetti verticali e persino *@-espressioni*. Normalmente il contenuto di `\multicolumn` rimpiazza completamente tutte le ⟨*numero*⟩ celle adiacenti, compresi i loro separatori destri; quindi il separatore sinistro non dovrebbe mai essere indicato come contenuto del ⟨*descrittore*⟩ a meno che il gruppo di celle non comprenda anche la prima cella di una riga.

`\hline` serve per tirare un filetto orizzontale attraverso tutta la tabella o tutta la matrice. Esso può essere eseguito solo come primo elemento di una tabella, oppure dopo il comando di fine riga `\\`, oppure dopo un altro comando `\hline`; in quest'ultimo caso vengono tracciati due filetti ravvicinati attraverso tutta la tabella.

`\cline{`⟨col_1-col_2⟩`}` serve per tracciare un filetto orizzontale solo sotto le colonne ⟨col_1-col_2⟩; per esempio in una tabella a 7 colonne, il comando `\cline{2-6}` traccia un filetto orizzontale sotto le 5 colonne centrali; non si possono inserire due identici comandi `\cline` uno di seguito all'altro per raddoppiare il filetto, ma si possono usare per 'sottolineare' colonne distinte: per esempio, nella tabella a 7 colonne dell'esempio precedente `\cline{2-3}\cline{5-6}` 'sottolinea' solo le colonne 2 e 3 con un filetto e, allineato con il primo, un secondo filetto 'sottolinea' le colonne 5 e 6.

I parametri compositivi di tabelle e matrici sono i seguenti.

`\arraycolsep` è metà dello spazio di separazione fra due celle consecutive di una matrice.

`\tabcolsep` e metà dello spazio di separazione fra due celle consecutive di una tabella.

`\arrayrulewidth` è lo spessore dei filetti orizzontali o verticali che si possono inserire in tabelle e matrici.

`\doublerulesep` è la spaziatura verticale o orizzontale fra due filetti consecutivi orizzontali o verticali.

`\arraystretch` è una macro, che può venire ridefinita con `\renewcommand`, e che rappresenta il fattore di scala con cui viene ingrandito o rimpicciolito lo strut che si trova sempre nella prima cella di ogni riga delle tabelle o delle matrici. Il suo valore di default è l'unità.

29.12 I file ausiliari e i loro comandi

29.12.1 I file del sistema TEX

LATEX, come anche Plain TEX, fa riferimento ad un gran numero di file dei quali spesso il compositore non si accorge nemmeno. A parte i file sorgente con estensione `.tex` o `.ltx`, ci sono anche i file con le estensioni seguenti; il nome proprio dei file corrispondenti ad un medesimo documento è costantemente il nome proprio del file principale e questo nome è conservato dentro la variabile `\jobname` che può essere usata anche dal compositore, o meglio, dal programmatore che scrive le macro.

`.aux` serve per conservare tutte le informazioni che riguardano i riferimenti incrociati; quando si usano i comandi `\include`, questi file conservano anche tutte le informazioni relative al file incluso in modo da poterne simulare la compilazione quando questo file non sia incluso nella lista di file da compilare contenuta nell'argomento di `\includeonly`. Il comando `\nofiles` sopprime la scrittura di tutti i file di questo tipo.

`.bbl` questo file viene scritto da BIBTEX, ma viene poi letto da LATEX per comporre la bibliografia quando si usi il comando `\bibliography`.

`.dvi` è il formato di default dell'uscita compilata di tutti i programmi elaborati con il sistema TEX; esso fa riferimento ai font a matrici di pixel e generalmente il programma per rendere leggibile agli umani questo file è incluso nella distribuzione del sistema TEX; pdflatex, che viene usato per produrre direttamente l'uscita in formato PDF, è in grado di produrre l'uscita in formato DVI se si pone all'inizio del file `.tex` la specificazione: `\pdfoutput=0`.

`.glo` è il file prodotto da LATEX quando sia attivo il comando `\makeglossary`; esso contiene tutte le voci `\glossaryentry` prodotte dai vari comandi `\glossary` inseriti nei file sorgente.

`.idx` è il file prodotto da LATEX quando sia attivo il comando `\makeindex`; esso contiene tutte le voci `\indexentry` prodotte dai vari comandi `\index` inseriti nel file sorgente.

`.ind` questo file viene scritto dal programma makeindex ma viene riletto da LATEX quando deve comporre l'indice analitico.

`.lof` contiene le informazioni per comporre l'indice delle figure.

`.log` contiene tutta le registrazione di quanto è successo durante l'esecuzione di LATEX, in particolare le informazioni relativamente dettagliate concernenti gli errori e gli avvertimenti, i font usati, quelli che sono stati creati al volo, quelli mancanti, quelli sostituiti, eccetera.

`.lot` contiene le informazioni per comporre la lista delle tabelle.

`.toc` contiene le informazioni per comporre l'indice generale.

29.12.2 I riferimenti incrociati

I comandi per assegnare una parola chiave agli oggetti da citare e/o per richiamarne il valore o la pagina sono:

```
\label{⟨chiave⟩}
\ref{⟨chiave⟩}
\pageref{⟨chiave⟩}
```

Il comando `\label` è fragile; sarebbe meglio, anche se è possibile, non inserirlo mai all'interno di altri comandi, nemmeno i comandi di sezionamento o i comandi per le didascalie. I contatori che possono essere citati con la chiave sono quelli che sono stati incrementati con `\refstepcounter`; in ogni caso `\label` associa alla chiave il valore dell'ultimo contatore in ordine di tempo che è stato incrementato con quel comando.

29.12.3 Bibliografia e citazioni

Salvo disporre di file bibliografici esterni e di fare uso del programma BibTeX, LaTeX dispone dell'ambiente *thebibliography* per comporre gli elenchi bibliografici e assegnare loro una chiave di riferimento.

```
\begin{thebibliography}{⟨cifre⟩}
\bibitem[⟨etichetta⟩]{⟨chiave⟩} ⟨voce bibliografica⟩
...
\end{thebibliography}
```

Le voci bibliografiche vengono richiamate con il comando

```
\cite[⟨testo⟩]{⟨chiave⟩}
oppure
\cite{⟨lista di chiavi⟩}
```

La ⟨*chiave*⟩ è una informazione interna di LaTeX che serve a LaTeX stesso e al compositore per riferirsi ad una particolare voce bibliografica. Quando questa viene citata, viene citata o con l'⟨*etichetta*⟩, se è stata specificata, oppure con il numero progressivo dell'elenco bibliografico; in ogni caso numero o etichetta vengono racchiusi entro parentesi quadre.

Con `\cite` si possono fare citazioni multiple, semplicemente specificando una ⟨*lista di chiavi*⟩ formata dalle chiavi specifiche dei riferimenti da citare separate da virgole; per questo motivo le chiavi possono essere formate con qualunque carattere, esclusa la virgola. Se si cita un'opera sola, la citazione contiene l'⟨*etichetta*⟩, o il numero, seguiti da ⟨*testo*⟩; per esempio: avendo attribuito ad un riferimento l'etichetta `TUG2007`, il comando di citazione `\cite[cap.~2]{TUG2007}` produrrà qualcosa come [12, cap. 2].

Per la bibliografia, specialmente se contiene numerose voci, è consigliabile servirsi dei programmi BibTeX o biber, che garantiscono la composizione uniforme delle varie citazioni e, tramite file di estensione, consente anche di usare stili di citazione diversi, come per esempio lo stile 'autore-anno'.

29.12.4 Suddivisione del file sorgente

Non è conveniente scrivere un unico file sorgente, specialmente se il documento da comporre contiene molte pagine, molti capitoli, eccetera. Conviene predisporre un 'master file' che contenga solo il preambolo e la lista dei file componenti.

```
\documentclass[⟨opzioni⟩]{⟨classe⟩}
\usepackage[⟨opzioni⟩]{⟨pacchetto⟩}[⟨data⟩]
...
\includeonly{⟨lista di file⟩}
\begin{document}
\include{⟨primo file⟩}
\include{⟨secondo file⟩}
...
\end{document}
```

La ⟨*lista di file*⟩ è un elenco di nomi di file separati da virgole, ma senza spazi spuri prima o dopo le virgole. La sequenza di comandi `\include`, ognuno con il suo nome di file, esegue la compilazione dei file, tranne di quelli che *non* sono elencati nella ⟨*lista di file*⟩. Al limite si può compilare un file alla volta; i file già compilati, anche se non vengono più elaborati, hanno lasciato le loro tracce nei rispettivi file `.aux`, quindi, a meno che non siano intervenute modifiche, LaTeX riesce a risolvere tutti i riferimenti incrociati e a mantenere la coerenza dei numeri delle pagine e di tutti i contatori specificati con i comandi specifici di LaTeX.

È chiaro che prima di terminare la lavorazione di un documento è importante eseguire la compilazione di tutto il documento, cosicché si è completamente sicuri che tutti i riferimenti e tutti i valori dei contatori e delle pagine siano corretti; questo lavoro, va fatto comunque per predisporre alla fine della lavorazione la compilazione dell'indice analitico.

Ci si ricordi che ogni comando `\include` per prima cosa esegue un comando `\clearpage` che svuota le code di oggetti flottanti e comincia la compilazione del nuovo file su una pagina nuova. Se *non* è questo quello che si vuole ottenere, si faccia uso del comando

```
\input{⟨file⟩}
```

che non consente la compilazione selettiva, ma consente di proseguire la compilazione del testo dal punto preciso nel quale si trovava LaTeX nel momento in cui ha cominciato ad eseguire il comando `\input`.

Per risolvere le dipendenze di un dato documento da certi file di macro, è disponibile l'unico ambiente che può precedere il comando `\documentclass`

```
\begin{filecontents}{⟨nome file⟩}
⟨corpo del file⟩
\end{filecontents}
```

È disponibile anche l'ambiente asteriscato; la differenza consiste in questo: quando *filecontents* viene eseguito, controlla che un file con il ⟨*nome file*⟩ specificato esista; se esiste, l'ambiente non fa nulla salvo scrivere sullo schermo e nel file `.log` che il file esiste già; se invece il file non esiste, questo ambiente lo crea scrivendovi dentro il ⟨*corpo del file*⟩ e salvandolo con il nome ⟨*nome file*⟩. Nel fare questo, al fine di identificare il tipo di file e la sua origine, esso scrive anche dei commenti in stile LaTeX, cioè con un segno `%` all'inizio della riga. Se non si vogliono questi commenti si usi l'ambiente asteriscato.

Il comando `\listfiles` serve per elencare a schermo e nel file `.log` tutti i nomi dei file che vengono via via aperti (tranne quelli di servizio) in modo che si possa seguire e controllare che l'esecuzione della compilazione proceda regolarmente.

29.13 Indice analitico e glossario

Come descritto nel capitolo 12 per comporre uno o più indici analitici è opportuno servirsi del programma makeindex; nello stesso capitolo viene anche descritto come servirsi di quel programma per compilare un glossario. Tuttavia qui si richiamano i comandi necessari.

29.13.1 Indice analitico

La composizione dell'indice analitico avviene mediante l'ambiente *theindex*

```
\begin{theindex}
\input{⟨indexfile⟩.ind}
\end{theindex}
```

Più comodamente si usa il pacchetto *makeidx* e si ordina la composizione dell'indice analitico mediante il comando `\printindex`. Ancora più comodamente si usa uno dei pacchetti *imakeidx* o *indextools* che consentono di sfruttare la proprietà dei sistemi TeX moderni di eseguire dei comandi di sistema così da poter generare uno o più indici analitici con una sola esecuzione di pdflatex.

Però per raccogliere le voci da indicizzare bisogna usare i seguenti comandi:

`\makeindex` inserito nel preambolo ordina di eseguire effettivamente la raccolta delle voci attraverso il comando `\index`, che altrimenti sarebbe completamente inerte.

`\index` serve per raccogliere le voci da indicizzare con la sintassi

```
\index{⟨voce⟩}
```

Il comando `\index` deve essere scritto senza spazi interposti alla fine della parola di testo che si vuole indicizzare. La ⟨*voce*⟩ non contiene solo la parola da indicizzare, ma anche le ulteriori informazioni di cui necessita il programma makeindex per formattare convenientemente la voce come voce principale, oppure secondaria, oppure di terzo livello, per scegliere il font con cui scrivere la voce, per scegliere la chiave di indicizzazione, per segnalare, nel caso di voci secondarie o di terzo livello, sotto quale voce primarie esse debbano essere elencate, per dare uno stile al numero di pagina, eccetera; si veda a questo proposito la documentazione del programma makeindex.

Chiaramente se si usano i pacchetti *imakeidx* o *indextools* per produrre diversi indici analitici bisogna specificare con un parametro facoltativo in quale indice si deve porre la voce particolare marcata con `\index`; per questo è meglio consultare la documentazione di quei pacchetti.

29.13.2 Glossario

Per comporre un glossario bisogna procedere in modo simile a quello che si usa per l'indice analitico; i comandi che permettono di raccogliere le voci sono

`\makeglossary` inserito nel preambolo consente l'effettiva raccolta delle voci mediante l'attivazione del comando `\glossary`, che altrimenti sarebbe totalmente inattivo.

`\glossary` serve per raccogliere le voci da inserire nel glossario, con la sintassi:

```
\glossary{⟨voce⟩}
```

29.14 Compilazione interattiva

LaTeX consente un certo livello di interattività durante la compilazione; a parte gli errori, che chiedono al compositore di intervenire, LaTeX consente anche di emettere dei messaggi sul terminale e di leggere le risposte del compositore.

`\typeout` con la sintassi

```
\typeout{⟨messaggio⟩}
```

consente di scrivere un messaggio sullo schermo del terminale; questo messaggio può avere gli scopi più diversi, ma non consente nessuna interattività; piuttosto può preparare un dialogo interattivo.

`\typein` con la sintassi

```
\typein[⟨comando⟩]{⟨messaggio⟩}
```

emette il ⟨*messaggio*⟩ sullo schermo; se è specificato anche il ⟨*comando*⟩ il programma si arresta e attende che l'operatore scriva qualche cosa sulla tastiera (presumibilmente quanto il ⟨*messaggio*⟩ suggerisce di scrivere fra una rosa di possibili scelte); non appena il compositore preme il tasto di fine riga, il testo introdotto viene assegnato al ⟨*comando*⟩, esattamente come se il tutto eseguisse le seguenti operazioni

`\typeout{`⟨*messaggio*⟩`}`
`\newcommand{`⟨*comando*⟩`}{`⟨*testo introdotto*⟩`}`

Il ⟨*comando*⟩ deve essere una valida sequenza di controllo, cioè una stringa esclusivamente letterale preceduta dal backslash, oppure un solo segno non letterale preceduto dal backslash, oppure un carattere attivo.

29.15 Interruzione di riga e di pagina

29.15.1 Interruzione di riga

I comandi

```
\linebreak[⟨numero⟩]
\nolinebreak[⟨numero⟩]
```

consentono o vietano, rispettivamente, di interrompere una riga. Se si specifica il ⟨*numero*⟩ la loro azione è rinforzata o indebolita a seconda del valore del numero stesso. Il numero 0 rappresenta un debole incoraggiamento ad operare, mentre il numero 4 rappresenta l'indicazione di eseguire incondizionatamente l'azione specificata; l'assenza del numero equivale a specificare il numero 4. Attenzione: `\linebreak` interrompe la linea ma lascia che LaTeX cerchi di giustificarla, per cui se la gomma interparola deve essere allargata troppo, viene emesso un messaggio di `Underfull \hbox`.

Una riga può venire interrotta anche con i comandi

```
\\[⟨spazio⟩]
\\*[⟨spazio⟩]
\newline
```

Essi permettono di interrompere una riga per andare decisamente a capo. Non viene eseguito nessun tentativo di giustificare la riga, nemmeno se manca poco al margine destro. La forma asteriscata impedisce un fine pagina subito dopo.

Per tutti questi comandi la possibilità di eseguire la cesura delle parole in fin di riga è essenziale. In certe circostanze non basta consentire la cesura, ma bisogna aumentare le tolleranze di composizione, cioè bisogna accontentarsi di una maggiore bruttezza di una o più righe. Ciò si ottiene con i comandi

```
\sloppy
\fussy
```

Il primo aumenta le tolleranze di composizione a valori enormi, che per LaTeX equivalgono ad infinito. Il secondo ripristina i valori normali.

29.15.2 Interruzione di pagina

I comandi

```
\pagebreak[⟨numero⟩]
\nopagebreak[⟨numero⟩]
```

Agiscono come `\linebreak` ma riferendosi alla pagina, non alla riga. Il ⟨*numero*⟩ ha lo stesso significato con 0 che indica in blando incoraggiamento ad eseguire, e 4 che ordina di eseguire incondizionatamente il comando.

Per l'interruzione di pagina non si può agire sulla cesura, ma si può temporaneamente allungare la pagina rispetto al normale:

```
\enlargethispage{⟨lunghezza⟩}
\enlargethispage*{⟨lunghezza⟩}
```

Il comando senza asterisco allunga semplicemente la pagina; per ovvi motivi ⟨*lunghezza*⟩ dovrebbe essere un multiplo intero di `\baselineskip`, dello scartamento normale con il font di default. Tuttavia il comando asteriscato cerca di allungare la pagina di quanto specificato, ma cerca anche di stringere il più possibile gli spazi bianchi verticali al fine di non allungare la pagina, ovvero di allungarla il meno possibile. Specialmente componendo sul verso e sul recto è importante che le pagine siano della stessa altezza; se i margini laterali sono consistenti, una riga in più in una delle due pagine si nota appena, tuttavia sarebbe meglio evitarlo. Il comando asteriscato potrebbe essere la soluzione se la pagina nella quale esso compare ha abbastanza spazi bianchi.

I comandi

```
\newpage
\clearpage
\cleardoublepage
```

chiudono la pagina corrente; gli ultimi due scaricano anche le eventuali code di oggetti flottanti ancora non svuotate; il comando `\cleardoublepage` eventualmente emette anche una pagina bianca al fine di consentire di iniziare la pagina successiva sul recto. `\cleardoublepage` è tipicamente il primo comando che viene eseguito implicitamente quando si esegue il comando di inizio di un nuovo capitolo e anche quando si immette un file con `\include`.

Sono disponibili anche dei comandi presi a prestito da Plain TeX, ma ancora definiti con LaTeX:

`\smallbreak` esegue un salto pagina solo se prima era stato introdotto nel testo uno spazio verticale pari a `\smallskipamount`.

`\medbreak` esegue un salto pagina solo se prima era stato introdotto nel testo uno spazio verticale pari a `\medskipamount`.

`\bigbreak` esegue un salto pagina solo se prima era stato introdotto nel testo uno spazio verticale pari a `\bigskipamount`.

`\goodbreak` indica a LaTeX che questo è un buon punto per spezzare la pagina; LaTeX effettivamente esegue il salto se manca poco alla fine naturale della pagina, se, cioè non resta una pagina pesantemente mozza, ma è solo corta di pochissime righe.

`\filbreak` sempre se manca poco alla fine della pagina, questo comando indica a LaTeX che può eseguire un salto pagina inserendo anche un po' di gomma verticale in modo che non vengano emessi messaggi di `Underfull vbox`; nello stesso tempo, se il punto non è buono per interrompere la pagina `\filbreak` non fa nulla.

Si consiglia di usare questi comandi solo alla fine della lavorazione di un documento per correggere a mano alcune piccole cose che LaTeX da solo non è capace di gestire. I casi in cui il compositore è costretto a ricorrere a questi comandi e a quelli di LaTeX sono rarissimi in testi 'normali'; possono essere frequenti in testi che contengono molta matematica. Tuttavia `\enlargethispage` con o senza asterisco consente di rimediare ad un capitolo la cui ultima riga o ultime due righe cadono nell'ultima pagina, peggio ancora se questa pagina è dispari. Se la pagina è pari questa eventualità non è tanto bella ma può essere accettata; se esse cadono su una pagina dispari, questa resta quasi bianca, e il capitolo successivo inizia in generale di nuovo su una pagina dispari, con il risultato che appaiono due pagine bianche o quasi bianche di seguito; decisamente da evitare.

Piuttosto se i capoversi finali del capitolo problematico sono sufficientemente lunghi si può convincere pdftex a comporli su una o due dighe di più o di meno, senza forzare allargamenti o accorciamenti improponibili degli spazi interparola, specificando il comando primitivo `\looseness` seguito da un (piccolo) numero positivo o negativo; per esempio scrivendo `\looseness=1` dentro o in assoluta adiacenza ad un capoverso, pdftex cercherà di allungare quel solo capoverso di una riga purché il farlo non aumenti la 'bruttezza' del capoverso oltre alla tolleranza valida in quel momento. Il valore di `\looseness` viene rimesso a zero automaticamente alla fine del capoverso, quindi vale solo per quel capoverso per il quale è stato specificato. Per esempio, questo capoverso è stato composto con un valore unitario di questo parametro.

29.16 Scatole

Una *scatola* è un oggetto che può contenere diversi segni, anche un testo formato da diverse righe, ma questo testo, una volta inscatolato, viene trattato da LaTeX

come se fosse un oggetto unico e non lo spezzerà mai alla fine di una riga o di una pagina.

29.16.1 Scatole di uso immediato

I comandi per mettere del ⟨*testo*⟩ dentro una scatola sono

```
\mbox{⟨testo⟩}
\makebox[⟨larghezza⟩][⟨posizione⟩]{⟨testo⟩}
\fbox{⟨testo⟩}
\framebox[⟨larghezza⟩][⟨posizione⟩]{⟨testo⟩}
\begin{lrbox}{⟨scatola⟩}⟨testo⟩\end{lrbox}
```

I comandi che cominciano con la lettera 'f' inseriscono una cornice attorno al testo; quelli che cominciano con la lettera 'm' non inseriscono nessuna cornice. Se la ⟨*larghezza*⟩, che è facoltativa, viene specificata, il testo viene inserito dentro una scatola della larghezza specificata, altrimenti i comandi che accettano questo argomento facoltativo si comportano come quelli che non lo richiedono. Se viene specificata la ⟨*larghezza*⟩ allora si può specificare anche la posizione mediante una sola delle lettere seguenti: `l` per collocare il testo a sinistra, `r` per collocare il testo a destra, `s` per distribuire il testo, allargando o restringendo lo spazio interparola, lungo tutta la ⟨*larghezza*⟩; se non viene specificata nessuna posizione il testo risulta centrato dentro la ⟨*larghezza*⟩. Si confrontino i casi seguenti.

```
\framebox[50mm][l]{testo a sinistra}
\framebox[50mm][r]{testo a destra}
\framebox[50mm]{testo centrato}
\framebox[50mm][s]{testo spaziato}
```

testo a sinistra

testo a destra

testo centrato

testo spaziato

Lo spessore della linea che forma la cornice delle scatole è conservata nel parametro dimensionale `\fboxrule` e la distanza della cornice dal testo è conservata dentro il parametro dimensionale `\fboxsep`. Attenzione: lo spessore di 1 pt per la cornice dà luogo a una cornice molto scura: testo. Quindi non è il caso di specificare spessori maggiori se non a ragion veduta.

Si può sostituire l'indicazione esplicita della⟨*larghezza*⟩ nelle scatole che l'accettano mediante quattro nuovi comandi che si riferiscono tutti alla larghezza della scatola composta con `\mbox` che contiene lo stesso ⟨*testo*⟩; esse sono:

`\height` ovvero l'altezza della scatola composta con `\mbox{`⟨*testo*⟩`}`
`\depth` ovvero la profondità di quella scatola
`\totalheight` ovvero la somma di `\height` e di `\depth`
`\width` ovvero la larghezza di quella scatola.

29.16.2 Scatole per conservare del testo

Oltre ad assegnare nomi simbolici alle scatole mediante il comando `\newsavebox`, LaTeX consente di memorizzare alcune scatole in appositi registri, e di usarne il contenuto in diversi modi; chiaramente le macro del mark-up di LaTeX sono definite in termini di comandi primitivi del sottostante linguaggio TeX, ma mentre questi comandi permettono di fare cose raramente necessarie per l'utente (il quale se sente la necessità di usarle è un utente abbastanza avanzato per essersi documentato sul TeXbook), le macro di LaTeX servono per eseguire le operazioni più frequenti; esse sono le seguenti:

`\newsavebox` con la sintassi

> `\newsavebox{`⟨*sequenza di controllo*⟩`}`

alloca un registro-scatola ed assegna il suo indirizzo alla ⟨*sequenza di controllo*⟩; questa sequenza deve essere conforme nella sua forma a tutte le sequenze di LaTeX quindi deve cominciare con una barra rovescia e proseguire con una stringa di sole lettere maiuscole o minuscole; è evidente che è bene che questa stringa abbia un valore mnemonico, cosicché l'utente sa dal nome a che scopo ha definito quella scatola.

`\sbox` con la sintassi

> `\sbox{`⟨*scatola*⟩`}{`⟨*testo*⟩`}`

carica la ⟨*scatola*⟩ con il ⟨*testo*⟩; si noti che ⟨*scatola*⟩ può essere tanto un nome assegnato ad un registro-scatola mediante `\newsavebox` (e questo varrà anche per i comandi successivi), ma anche direttamente un indirizzo assoluto di un registro-scatola. Questa seconda possibilità è appunto una possibilità, ma è fortemente sconsigliabile, perché specificando un indirizzo "sbagliato" si rischia di caricare un registro-scatola usato dal nucleo di LaTeX, cosa che potrebbe produrre errori difficilmente diagnosticabili.

Attenzione: va detto qui, ma lo si potrebbe ripetere anche per i prossimi comandi: il processo di allocazione dei registri-scatola mediante `\newsavebox` permette di associare una ⟨*sequenza di controllo*⟩ con gli indirizzi fino al 254; il registro 255 è riservato per il nucleo di LaTeX. Per allocare e nominare altri registri scatola bisogna caricare il pacchetto ***etex***; però se si vogliono usare gli indirizzi assoluti da 256 in su fino a 32 767, è possibile farlo senza necessità di caricare quel pacchetto; ma è pericoloso e lo si sconsiglia fortemente.

Si noti ancora che `\sbox` funziona in modo simile a `\mbox` solo che quest'ultimo non carica la scatola in memoria ma la usa subito per il testo che si sta componendo.

`\savebox` continua la similitudine di questo comando con quello di `\makebox`; esso ha la sintassi:

`\savebox{`⟨*scatola*⟩`}[`⟨*larghezza*⟩`][`⟨*posizione*⟩`]{`⟨*testo*⟩`}`

ed agisce come `\makebox`, a parte la memorizzazione del contenuto in un registro-scatola, e i suoi argomenti obbligatori e facoltativi hanno gli stessi significati.

`\usebox` con la sintassi:

`\usebox{`⟨*scatola*⟩`}`

permette di usare il contenuto della ⟨*scatola*⟩ ma questo contenuto rimane memorizzato nel suo registro ed è usabile con successivi comandi `\usebox`; tuttavia se il contenuto della ⟨*scatola*⟩ è stato assegnato mentre si era dentro un gruppo o dentro un ambiente, come si esce dal gruppo o dall'ambiente il contenuto del registro viene ripristinato al valore che esso aveva prima di aprire il gruppo o l'ambiente.

29.16.3 Ambienti e comandi per scatole particolari

L'ambiente *lrbox* è un po' particolare; ha la sintassi seguente:

```
\begin{lrbox}{⟨scatola⟩}
⟨testo⟩
\end{lrbox}
```

e serve per comporre il ⟨*testo*⟩ dentro la ⟨*scatola*⟩ specificata con il suo numero o il suo nome simbolico precedentemente dichiarato mediante il comando `\newsavebox`; in un certo senso esso si comporta come se venissero eseguiti i comandi primitivi:

`\setbox`⟨*scatola*⟩`\hbox{`⟨*testo*⟩`}`

ma in realtà fa molto di più: infatti ⟨*testo*⟩ può essere qualunque cosa che abbia senso in modo orizzontale, compreso, per esempio, un ambiente *minipage* (vedi poco più avanti); il fatto che non sia necessario usare le graffe per delimitare il ⟨*testo*⟩, rende questo ambiente particolarmente utile per definire ambienti che ricorrono a comandi i cui argomenti debbano essere delimitati da graffe vere, non sostituibili con graffe logiche. Per esempio se si volesse comporre un testo di diverse righe composto dentro una cornice, non lo si potrebbe fare direttamente con il comando `\fbox` ma lo si potrebbe fare definendo un ambiente *riquadro* in questo modo:

```
\newenvironment{riquadro}[1][\linewidth]{% apertura
\begin{lrbox}{0}\begin{minipage}{\dimexpr#1-2\fboxsep-2\fboxrule}
}{% chiusura
\end{minipage}\end{lrbox}\fbox{\usebox{0}}\relax
}
```

e questo nuovo ambiente verrebbe usato così:

```
\begin{riquadro}[⟨larghezza⟩]
⟨testo⟩
\end{riquadro}
```

Per esempio:

Questo è un testo riquadrato mediante l'ambiente *riquadro* definito poco sopra.

Il procedimento si basa sul fatto che l'ambiente *lrbox* carica una scatola con una "paginetta" di giustezza specificata (in mancanza della specificazione dell'argomento facoltativo viene usata la giustezza corrente del testo, `\linewidth`) composta con *minipage* (vedi poco oltre), poi, dopo aver finito questo caricamento, usa la scatola come argomento di `\fbox` che le disegna la cornice attorno.

Per inserire del testo dentro scatole che contengono materiale 'verticale' (in pratica una successione verticale di righe, uno o più capoversi) si può usare il comando `\parbox` o l'ambiente *minipage*:

```
\parbox[⟨allineamento⟩][⟨altezza⟩][⟨pos.interna⟩]{⟨giustezza⟩}{⟨testo⟩}
```

oppure

```
\begin{minipage}[⟨allineamento⟩][⟨altezza⟩][⟨pos.interna⟩]{⟨giustezza⟩}
⟨testo⟩
\end{minipage}
```

dove il parametro ⟨*allineamento*⟩ indica l'allineamento della scatola con il testo circostante; `t` indica che la riga di base della prima riga della scatola è allineata con la riga di base del testo circostante; `b` indica che la linea di base dell'ultima riga della scatola è allineata con la riga di base del testo circostante; con il codice `c` predefinito, e quindi senza specificare nulla, l'asse matematico della scatola è allineato con l'asse matematico del testo circostante. Facoltativamente si può specificare l'⟨*altezza*⟩ della scatola; se questa altezza viene specificata, allora si può usare il terzo parametro facoltativo ⟨*pos.interna*⟩; le lettere di posizionamento sono le stesse di quelle usate per l'allineamento dell'intera scatola, ma si riferiscono a dove il testo contenuto nella scatola stessa viene collocato verticalmente rispetto ai suoi bordi ideali. Va infine specificata la ⟨*giustezza*⟩ della scatola, in pratica la sua larghezza, e il ⟨*testo*⟩ che essa deve contenere.

La differenza sostanziale fra il comando e l'ambiente è la seguente: il comando `\parbox` non comincia a comporre il ⟨*testo*⟩ al suo interno, finché non l'ha

letto tutto. Tutto ciò non avviene con l'ambiente *minipage* il quale gestisce il ⟨*testo*⟩ in modo differente. In compenso *minipage* in quanto ambiente, richiede una 'amministrazione' un poco più complessa che richiede un maggior tempo di elaborazione. Parlando di microsecondi, non è il caso di formalizzarsi troppo, visti i numerosissimi vantaggi che offre l'ambiente rispetto al comando. L'ambiente *minipage* può anche contenere delle note interne alla 'paginetta' che esso compone; queste note vengono composte con i soliti comandi; solo che la numerazione delle note avviene con lettere corsive in posizione di apice, in modo da distinguerle bene dalle note di piè di pagina, generalmente composte con esponenti numerici o con simboli non letterali.

Si noti che se il contenuto di una *minipage* comincia con una equazione in display, essa conserva il spazio che precede l'equazione; se non si vuole questo spazio si cominci la *minipage* con `\vspace{-\abovedisplayskip}`.

Il comando `\rule` compone una scatola particolare: essa, infatti, è completamente nera. La sintassi è:

`\rule[`⟨*rialzo*⟩`]{`⟨*base*⟩`}{`⟨*altezza*⟩`}`

La ⟨*base*⟩ e l'⟨*altezza*⟩ sono la base e l'altezza del rettangolo nero; se una delle due è nulla il rettangolo è invisibile, ma l'oggetto continua a mantenere l'altra dimensione; se la base è nulla si ottiene uno *strut*, che in gergo tipografico italiano si chiama *pilastrino*. In questo testo si è mantenuto il nome inglese per ricordare più facilmente i nomi dei comandi che inseriscono *strut* vari sia nelle tabelle, sia in matematica. Il rettangolo nero (o invisibile) risulta rialzato di ⟨*rialzo*⟩ se questa lunghezza è positiva, o ribassato se essa è negativa.

Infine un comando serve per collocare del ⟨*testo*⟩ in una scatola che viene alzata o abbassata a piacere e le si possono assegnare dimensioni a piacere, indipendentemente dal testo che la scatola contiene.

`\raisebox{`⟨*rialzo*⟩`}[`⟨*altezza*⟩`][`⟨*profondità*⟩`]{`⟨*testo*⟩`}`

La ⟨*profondità*⟩ può venire specificata solo se si specifica anche l'⟨*altezza*⟩.

29.16.4 Cenno ai comandi per le scatole con il linguaggio primitivo di TEX

Come accennato sopra, il linguaggio primitivo di TEX, si cui si basano le macro del mark-up di LATEX, sono più numerose e svolgono anche altri compiti che con i comandi di LATEX non si possono svolgere. La descrizione che segue è molto sommaria; per approfondire bisogna consultare il TEXbook.

`\hbox` è simile a `\mbox` ma se viene usato in modo verticale il modo di composizione non viene modificato, mentre con `\mbox` si passa al modo orizzontale. L'argomento di `\hbox` può anche venire racchiuso fra graffe simboliche, per esempio `\bgroup` e `\ergoup`, cosicché nella definizione di ambienti la graffa simbolica di apertura può stare fra i comandi di apertura e la

graffa simbolica di chiusura può stare fra i comandi di chiusura, senza che siano strettamente appaiate all'interno dei due argomenti di definizione del comando `\newenvironment`. la sua sintassi è sostanzialmente uguale a quella di `\mbox`, ma in linguaggio TeX è ammessa una variante

`\hbox to`⟨*larghezza*⟩`{`⟨*testo*⟩`}`

che si comporta come `\makebox`; la posizione del testo dentro la scatola viene definito specificando a sinistra e a destra del ⟨*testo*⟩ della "gomma" sufficiente per rispettare la ⟨*larghezza*⟩ specificata; normalmente questa "gomma" viene specificata con `\hss` il cui nome significa *horizontal stretch and shrink*; si tratta di una lunghezza elastica di lunghezza naturale nulla ma con allungamento e accorciamento illimitati.

`\vbox` compone una scatola verticale come farebbe `\parbox[b]`, ma il suo argomento può essere delimitato da graffe simboliche.

`\vtop` compone una scatola verticale come farebbe `\parbox[t]`, ma il suo argomento può essere delimitato da graffe simboliche.

`\vcenter` compone una scatola verticale come farebbe `\parbox[c]` o quando non è specificata nessuna opzione di posizionamento, ma il suo argomento può essere delimitato da graffe simboliche. Può essere usato solo in modo matematico, visto che il suo allineamento verticale è basato sull'asse matematico della scatola allineato con l'asse matematico del testo circostante.

Tutti e tre questi comandi ammettono le varianti `\vbox to`. `\vtop to` e `\vcenter to`, ma la "gomma" deve essere ora specificata con `\vss` (*vertical stretch and shrink*) per collocare il ⟨*testo*⟩ in alto o in basso o al centro della lunghezza (altezza) specificata.

`\newbox` del tutto simile a `\newsavebox`.

`\setbox` è simile a `\sbox` per le scatole orizzontali di LaTeX, ma in TeX serve anche per le scatole verticali; la sintassi è:

`\setbox`⟨*scatola*⟩⟨*comando scatola*⟩`{`⟨*testo*⟩`}`

dove ⟨*scatola*⟩ rappresenta l'indirizzo assoluto di un registro-scatola oppure il nome simbolico di una scatola allocata mediante `\newbox`; il ⟨*comando scatola*⟩ è uno dei comandi TeX per inscatolare il testo: `\hbox`, `\vbox`, `\vtop`, `\vcenter` (questo con qualche cautela a causa dell'ambiente matematico necessario per usarlo).

`\box` serve per inserire la ⟨*scatola*⟩ identificata dal suo argomento nel testo che si sta componendo; il corrispondente registro-scatola viene svuotato. La sintassi per questo e i comandi successivi è

`\box`⟨*scatola*⟩

dove ⟨*scatola*⟩ è l'indirizzo assoluto di un registro-scatola oppure il suo nome simbolico.

\unhbox serve per "disinscatolare" il ⟨*testo*⟩ contenuto nella ⟨*scatola*⟩; dopo aver usato questo comando il registro-scatola è vuoto; la differenza rispetto a `\box` è che quest'ultimo non estrae il contento dalla scatola, ma lascia il ⟨*testo*⟩ inscatolato e gestibile come un unico oggetto; con `\unhbox` il singoli token contenuti nella scatola orizzontale vengono estratti e possono venire ricomposti a piacere. Un esempio tipico di questa differenza è contenuto nella definizione standard della didascalia con le classi standard; in questo caso, infatti, l'argomento obbligatorio di `\caption` che corrisponde al testo (lungo) della didascalia viene memorizzato in una scatola; viene misurata la larghezza di tale scatola e se è inferiore alla giustezza del testo corrente, la scatola viene estratta in una riga centrata; altrimenti il contenuto della scatola viene estratto e ricomposto in una scatola verticale `\vtop` (per allineare la prima riga della didascalia con il suo titolino) con la giustezza specifica per le didascalie e poi la scatola verticale viene copiata dove va composta la didascalia. Ovviamente l'utente non si accorge di tutto questo lavorio dietro le quinte, e questo è uno dei pregi del sistema di macro che costituiscono il make-up di LaTeX.

\unvbox serve per "disinscatolare" le scatole orizzontali (le righe) che costituiscono il contenuto di una scatola verticale; quindi non si estrae il testo immesso nella scatola, ma si estraggono le righe nelle quali è stato composto il testo. Alla fine dell'operazione la scatola è vuota.

\unhcopy agisce come `\unhbox` ma non svuota il registro; si comporta in modo simile al comando di LaTeX `\usebox`.

\unvcopy agisce come `\unvbox` ma non svuota il registro.

\vsplit serve per estrarre una parte del contenuto di una scatola verticale estraendola dal registro originale; si usa sostanzialmente così:

`\setbox`⟨*scatola2*⟩ `=\vsplit` ⟨*scatola1*⟩ `to` ⟨*dimensione*⟩

dove ⟨*scatola1*⟩ è una scatola verticale che contiene un certo numero di righe (scatole orizzontali) di altezze variabili se sono state composte con font di corpi diversi; ⟨*dimensione*⟩ non è un numero, ma è l'altezza complessiva delle righe che si vogliono estrarre da ⟨*scatola1*⟩ per metterle in ⟨*scatola2*⟩. Tipicamente questa è l'operazione eseguita dalla routine di uscita di LaTeX quando deve estrarre una pagina composta da spedire al file di uscita prelevando la parte superiore della scatola-registro 255, riservato da LaTeX per accumulare via via i capoversi del testo immesso

dall'utente. Ovviamente la routine di uscita non si limita ad estrarre il testo dalla scatola 255, per comporre la pagina in uscita, visto che deve inserire gli oggetti mobili, le note, le testatine e i piedini, insomma, visto che deve confezionare la pagina completa di tutto quello che occorre, non solo del testo.

29.16.5 Operazioni di misura sulle scatole

Le scatole possono servire a molti scopi; esse sono un ingrediente essenziale del modo di funzionare dei programmi di composizione del sistema TeX, ma esse possono venire usate dall'utente in vari modi.

Per esempio esse possono servire per pretendere le misure dell'ingombro di certi oggetti da inserire nel documento; basta metterli in una scatola e chiedere al programma di fornirne le misure; i comandi di LaTeX che si sono già descritti sono `\settowidth`, `\settoheight` e `\settodepth`, i cui argomenti sono un nome simbolico di una registro-dimensione e un testo o comunque un oggetto da misurare. Per non usare inutilmente registri interni del programma, si può fare anche in un altro modo, se si deve poi usare l'oggetto in un secondo tempo; basta memorizzare l'oggetto in una scatola, poi si usano i comandi di TeX`\ht`, `\wd` e `\dp` per misurare rispettivamente l'altezza, la larghezza e la profondità; sono sostanzialmente le stesse operazioni che svolgono i comandi di LaTeX, ma ognuno ogni volta prende il testo e lo memorizza in una scatola; si esamini questo esempio, dove vogliamo misurare le dimensioni del logo GuIT nel formato `\Huge`, ma volgiamo poter utilizzare questo grande logo più volte senza doverlo ricomporre:

```
% nel preambolo
\newsavebox{\scatolaGuIT}
\savebox{\scatolaGuIT}{\Huge \GuIT}}

% Nel corpo del documento
\dots\
Il logo di \GuIT\ in formato \texttt{\textbackslash Huge},
\usebox{\scatolaGuIT}, ha le dimensioni seguenti:
altezza \the\ht\scatolaGuIT; larghezza \the\wd\scatolaGuIT;
profondità \the\dp\scatolaGuIT.
\dots
```

che permette di ottenere:

> ...
> Il logo di GuIT in formato `\Huge`, GUIT, ha la dimensioni seguenti:
> altezza 17.00131pt; larghezza 41.97949pt; profondità 7.64856pt.
> ...

Il commando `\the` è già stato descritto nella pagina 661.

Esistono altri comandi nel linguaggio TEX per fare cose ancora più insolite con le scatole, ma si ritiene che quanto esposto sia sufficiente per dare una idea di che cosa il linguaggio TEX sia capace. Il lettore interessato deve documentarsi convenientemente sul TEXbook, che è la fonte primaria di informazione sul linguaggio TEX.

29.17 Disegni e colori

29.17.1 Disegni

Come già specificato l'ambiente *picture* nativo di LATEX non è in grado di fare altro che semplici disegni, con l'inclinazione dei segmenti e dei vettori molto limitata dagli speciali font che servono per disegnarli; anche i cerchi sia pieni sia vuoti sono un problema, se non nelle situazioni più semplici; i rettangoli ad angoli arrotondati soffrono delle stesse limitazioni dei cerchi; le linee arbitrarie possono essere eseguite solo con le curve di Bézier quadratiche, quindi con certe limitazioni sulle quali qui non è il caso di insistere. Dal 2003 è disponibile il pacchetto *pict2e* che toglie tutte queste limitazioni; esso ha assunto una conformazione stabile dal 2004; quindi si raccomanda caldamente di usare quel pacchetto se si desiderano eseguire semplici disegni. Nel 2009 il pacchetto *pict2e* è stato ulteriormente arricchito di altri comandi che permettono ora di comporre disegni piuttosto elaborati.

Se si vuole disegnare qualcosa di più elaborato ancora, si ricorra al pacchetto *pgf* e al suo ambiente *tikzpicture*; questo pacchetto consente di disegnare praticamente qualsiasi cosa. Se non dovesse bastare c'è sempre il pacchetto *PSTricks* che è limitato solo dal fatto che il linguaggio PostScript su cui si basa non è onnipotente, ma quasi.

Qui, pertanto, non si darà nessuna informazione sull'ambiente *picture*; gli esempi svolti nel testo dovrebbero essere autoesplicativi. Si invita invece il lettore a documentarsi sui pacchetti *pict2e*, *pgf* e sul suo ambiente *tikzpicture*, nonché sul pacchetto *PSTricks*.

29.17.2 Colori e grafica

LATEX è dotato di una serie di pacchetti che consentono di eseguire diverse cose: oltre all'importazione di file grafici esterni, essi mettono a disposizione la gestione del colore e la manipolazione di scatole varie, che normalmente non possono essere eseguite da LATEX (se non passando attraverso il formato PostScript) ma possono invece venire eseguite direttamente da pdfLATEX. L'intera serie di pacchetti si trova nella cartella `.../tex/latex/graphics/`, mentre la documentazione si trova in `.../doc/latex/graphics/`. Il lettore è invitato a documentarsi sul file `grfguide.pdf`; qui si riportano solo i comandi più importanti.

Si noti che fra questi pacchetti c'è *color* per la gestione del colore; i colori possono essere chiamati per nome o possono essere descritti a livello più basso mediante la specificazione del modello del colore e le componenti dalle quali in questo modello un dato colore è caratterizzato. Il pacchetto *xcolor* estende moltissimo queste funzionalità. La gestione delle scatole e l'importazione di figure esterne è gestita da altri pacchetti: di questi si è citato qui sempre e soltanto *graphicx*, che è il più comodo da usare, in quanto esso dispone di una interfaccia a parole chiave, rispetto al file che contiene le macro vere e proprie di gestione della grafica, *graphics*. Quest'ultimo può essere usato da solo, ma la sintassi per fargli eseguire le cose che sono così semplici con *graphicx* diventa spesso abbastanza complessa.

`\scalebox{`⟨*scala-orizz.*⟩`}[`⟨*scala-vert.*⟩`]{`⟨*testo*⟩`}` serve per inserire ⟨*testo*⟩ dentro una scatola per poi scalarla del fattore di scala orizzontale; se il fattore di scala verticale non viene specificato l'ingrandimento verticale avviene con lo stesso fattore di scala orizzontale, altrimenti la scatola viene deformata in maniera diversa così da ottenere effetti particolari.

`\resizebox{`⟨*larghezza*⟩`}{`⟨*altezza*⟩`}{`⟨*testo*⟩`}` esegue in pratica la stessa operazione di `\scalebox` solo che invece di specificare dei fattori di scala, vengono specificate le dimensioni finali effettive. Il comando asteriscato agisce in modo da scalare l'altezza totale (altezza più profondità) al valore specificato. Una delle due dimensioni può essere sostituita da un punto esclamativo; in questo caso l'altra dimensione fissa il fattore di scala complessivo in modo che il rapporto di forma non sia modificato.

`\rotatebox[`⟨*lista di chiavi*⟩`]{`⟨*angolo*⟩`}{`⟨*testo*⟩`}` serve per ruotare la scatola che contiene il ⟨*testo*⟩ di un ⟨*angolo*⟩ specificato (in gradi) in senso antiorario; il ⟨*testo*⟩ può essere qualunque cosa: del testo vero e proprio, una figura, una tabella, ...

Nella ⟨*lista di chiavi*⟩ si può anche specificare mediante *origin* il punto attorno al quale eseguire la rotazione; si possono definire delle coordinate, ma ci sono alcuni punti speciali che possono essere rappresentati mediante codici letterali: `t`, `b`, `r`, `l` e `B` rappresentano le solite iniziali di 'top', 'bottom', 'right', 'left', ma `B` è un codice nuovo rispetto a quelli già incontrati altre volte: esso specifica la linea di 'Base', la 'Baseline', la linea, cioè, che attraversa orizzontalmente la scatola in corrispondenza del punto di riferimento; queste lettere sono da usare a due a due; per esempio: `tl` per indicare l'angolo in alto a sinistra, `Br` per indicare l'intersezione della linea di base con il lato destro della scatola, eccetera. Si osservino le tre diverse posizioni che assume la lettera 'Q' quando la si ruoti di 90° attorno ai tre possibili punti di rotazione di sinistra: '`bl`' , '`Bl`' e '`tl`' . Generalmente per ruotare una scritta in verticale, per esempio per metterla a fianco di un asse verti-

cale in un disegno, è conveniente eseguire la rotazione attorno al punto `Bl`.

`\reflectbox{`⟨*text*⟩`}` gira la scatola che contiene il ⟨*testo*⟩ attorno al suo asse verticale in modo che appaia come riflessa allo specchio

`\includegraphics[`⟨*lista di chiavi*⟩`]{`⟨*file grafico*⟩`}` serve per inserire una scatola che contiene l'immagine descritta nel ⟨*file grafico*⟩ elaborata secondo le numerose chiavi che si possono specificare, ognuna con il suo valore; se si specifica il *bounding box*, il rettangolo 'circoscritto', la versione asteriscata di questo comando permette di tagliare tutto quanto sporge fuori da detto rettangolo. Per questo scopo talvolta è meglio specificare le coordinate del `viewport` o le dimensioni delle strisce da ritagliare con `trim` e poi specificare la chiave `clip`; in questo modo si può avere un controllo migliore di quanto si sta eseguendo.

`\definecolor{`⟨*nome*⟩`}{`⟨*modello*⟩`}{`⟨*valore*⟩`}` serve per dare un ⟨*nome*⟩ ad un colore specificato mediante il suo ⟨*modello*⟩ e il ⟨*valore*⟩ dei parametri che in quel modello identificano il colore scelto. LaTeX considera come modelli di colore i seguenti:

`gray` è il modello di colore grigio identificato da un solo valore, un numero decimale nell'intervallo $0-1$; 0 vuol dire nero e 1 significa bianco; ogni valore intermedio specifica una certa gradazione di grigio.

`rgb` è il modello di colore degli schermi: rosso, verde, blu. I colori sono additivi e il valore che specifica un dato colore in questo modello è dato da tre numeri decimali separati da virgole, tutti e tre nell'intervallo $0-1$; 0 vuol dire 'assenza' di quella particolare componente di colore, 1 significa che quella componente di colore è totalmente satura. Per esempio $1,0,0$ indica il rosso saturo.

`cmyk` è il modello della quadricromia a stampa con i colori sottrattivi; ogni colore è identificato da quattro componenti corrispondenti ciascuna alle componenti ciano (celeste), magenta (lilla), giallo, nero. Ogni componente è pesata con un numero decimale nell'intervallo $0-1$ e la quaterna di valori decimali costituisce una lista di valori separati da virgole.

Con ogni modello di colore, sono predefiniti i seguenti colori: `white`, `black`, `red`, `green`, `blue`, `yellow`, `cyan`, `magenta`. Il driver specifico a seconda se si passi attraverso il programma `dvips` oppure se si proceda con pdfLaTeX, può definire altri colori. I nomi dei colori predefiniti con il driver `dvips` sono contenuti nel file `dvipsnam.def` in `.../tex/latex/graphics/`.

`\color{`⟨*nome*⟩`}` dichiara che da questo momento in poi si userà il colore con il ⟨*nome*⟩ specificato. La sua azione viene delimitata dai gruppi o dagli ambienti.

`\textcolor{`⟨*nome*⟩`}{`⟨*testo*⟩`}` serve per colorare il ⟨*testo*⟩ con il colore denominato ⟨*nome*⟩.

`\colorbox{`⟨*sfondo*⟩`}{`⟨*testo*⟩`}` agisce come `\fbox`, ma il ⟨*testo*⟩ viene composto su uno sfondo colorato con il colore di nome ⟨*sfondo*⟩.

`\fcolorbox{`⟨*bordo*⟩`}{`⟨*sfondo*⟩`}{`⟨*testo*⟩`}` agisce come `\colorbox` salvo che la cornice della scatola non è nera ma colorata con il colore di nome ⟨*bordo*⟩. I parametri `\fboxsep` e `\fboxrule` che definiscono lo spazio di separazione fra bordo e ⟨*testo*⟩ e lo spessore del bordo, sono gli stessi che regolano le caratteristiche della scatola incorniciata con `\fbox`.

`\pagecolor{`⟨*nome*⟩`}` questo comando globale assegna alle pagine intere uno sfondo con il colore denominato ⟨*nome*⟩ e questo colore permane finché non si dichiara un colore differente (eventualmente `white`).

29.18 Selezione dei caratteri

I comandi standard di LaTeX che hanno a che vedere con la scelta dei font verranno descritti qui di seguito. Per gestire i font in modo più professionale, anzi, per caricare altre collezioni di font e per gestirle al meglio, è necessario documentarsi nella guida `fntguide.dvi`.

29.18.1 Scegliere famiglia, forma e serie

I comandi per scegliere la forma e/o la serie da cui trarre i font per la composizione sono raccolti nella tabella 29.4; nella tabella 29.5 sono invece appaiate le dichiarazioni e i corrispondenti comandi.

Le dichiarazioni non possono essere usate in matematica; i comandi possono essere usati in matematica, e sono anche soggetti alla variazione automatica di corpo che compete agli indici e ai pedici di primo e di secondo livello *se e solo se* viene caricato anche il pacchetto *amsmath*, ma i risultati possono essere pessimi se i font matematici e quelli testuali non si accordano; per cui se ne sconsiglia l'uso e si raccomanda invece di usare il corretto comando matematico `\mathrm` per i pedici e gli apici che normalmente contengono solo una parola o una abbreviazione, sempre scritti senza accenti. Si può ricorrere anche al comando `\text` fornito dal pacchetto *amsmath* e al corrispondente comando `\intertext` per intercalare del testo ad espressioni matematiche incolonnate. Nell'argomento di questi due comandi, se fosse necessario, si può ricorrere tanto ai comandi quanto alle dichiarazioni. Vale la pena di ricordare che tanto `\text` quanto `\intertext` usano il font testuale, non il tondo matematico o simili altri font matematici. Ci si ricordi quindi della necessità di scegliere i font testuali

`\mdseries`	Serie media	`\upshape`	Forma diritta
`\bfseries`	**Serie nera**	`\itshape`	*Forma corsiva*
`\rmfamily`	Tondo con grazie	`\slshape`	*Forma inclinata*
`\sffamily`	Lineare senza grazie	`\scshape`	MAIUSCOLETTO
`\ttfamily`	`Monospaziato`	`\normalfont`	Font di default

Tabella 29.4: Dichiarazioni per la scelta di famiglia, serie e forma

`\textmd{`⟨*testo*⟩`}`	`\mdseries`	`\textup{`⟨*testo*⟩`}`	`\upshape`
`\textbf{`⟨*testo*⟩`}`	`\bfseries`	`\textit{`⟨*testo*⟩`}`	`\itshape`
`\textrm{`⟨*testo*⟩`}`	`\rmfamily`	`\textsl{`⟨*testo*⟩`}`	`\slshape`
`\textsf{`⟨*testo*⟩`}`	`\sffamily`	`\textsc{`⟨*testo*⟩`}`	`\scshape`
`\texttt{`⟨*testo*⟩`}`	`\ttfamily`	`\textnormal{`⟨*testo*⟩`}`	`\normalfont`

Tabella 29.5: Corrispondenza fra comandi e dichiarazioni

`\tiny`	`\scriptsize`	`\footnotesize`	`\small`	`\normalsize`
`\large`	`\Large`	`\LARGE`	`\huge`	`\Huge`

Tabella 29.6: Dichiarazioni di corpo

e quelli matematici in modo che si accordino gli uni con gli altri; di default i font CM, EC, LM, cm-super, ed altre simili collezioni di font, si accordano sia in modo testo sia in modo matematico.

29.18.2 Scegliere il corpo

La dimensione dei caratteri è legata a nomi che fanno riferimento al corpo 'normale', quello, cioè, usato di default; per i corpi inferiori a 10 pt la successione dei corpi procede di 1 pt in 1 pt; per i corpi superiori si procede con una successione geometrica di ragione 1,2; l'opzione di classe *11pt* corrisponde ad un corpo di 10,95 pt, ma il numero magico 10,95 altro non è che $10 \times \sqrt{1{,}2}$. I comandi per la scelta dei corpi relativi a quello normale sono raccolti nella tabella 29.6. Questi comandi non possono venire usati in matematica.

29.18.3 Corpi testuali e matematici

Si ricorda che fra i comandi disponibili (ma raramente usati) è disponibile il comando per associare i quattro corpi che servono per il testo e gli indici superiori e inferiori di primo e di secondo ordine per la matematica:

`\DeclareMathSizes{`⟨*testo*⟩`}{`⟨*textsize*⟩`}{`⟨*subscriptsize*⟩`}{`⟨*subsubscriptsize*⟩`}`

dove al corpo testuale ⟨*testo*⟩ vengono associati i tre corpi per la matematica: ⟨*textsize*⟩ per i simboli delle espressioni, ⟨*subscriptsize*⟩ per gli indici e i pedici di primo livello e ⟨*subsubscriptsize*⟩ per gli indici e i pedici di secondo livello.

Queste dichiarazioni sono più utili di quanto si pensi, perché talvolta nelle classi standard è desiderabile apportare qualche correzione, specialmente per i corpi più piccoli e più grandi.

29.18.4 Simboli speciali

Qualunque segno di qualunque font può venire stampato prescindendo dalle sue caratteristiche particolari; il comando

`\symbol`⟨*indirizzo*⟩

permette di accedere a qualunque simbolo di qualunque polizza di caratteri mediante il suo ⟨*indirizzo*⟩; il carattere viene stampato prescindendo dalle sue caratteristiche speciali o dall'esistenza di un comando specifico per usarlo; l'unica questione è conoscerne l'indirizzo decimale, oppure ottale, oppure esadecimale. Attenzione: gli indirizzi si riferiscono alla polizza di font usati per la composizione, non ai caratteri introdotti o introducibili con la tastiera; per esempio le due parentesi graffe nell'encoding di entrata hanno indirizzi 123 e 125 rispettivamente: se però si usano questi indirizzi con un font con codifica OT1, si ottiene – e ″, il che corrisponde perfettamente con quanto si può leggere nella tabella 18.3 della pagina 355.

Questo fatto è collegato al modo diverso di interpretare i caratteri immessi con la tastiera nel file sorgente `.tex` e soggetti alla codifica di entrata specificata con l'opzione fornita al pacchetto *`inputenc`* rispetto alla codifica usata per i font di uscita, specificata con l'opzione passata al pacchetto *`fontenc`*. Si veda più dettagliatamente questa questione nel capitolo 26.

Bibliografia

[1] Artech House. "Style Manual". Artech House, Boston – London.

[2] A.V. *The Chicago Manual of Style*. The University of Chicago Press, 14 ed., 1994.

[3] A.V. "Una (mica tanto) breve introduzione a LaTeX 2ε Ovvero LaTeX 2ε in 93 minuti". PDF document, 9 2000. In `$TEXMF/doc/latex/lshort-italian/itlshort.pdf`. Traduzione a cura di G. Agostini, G. Bilotta, F. Casadei Della Chiesa, O. de Bari, G. Delre, L. Ferrante, T. Pecorella, M. Rigido, R. Zanasi.

[4] Claudio Beccari. *Il LaTeX Refernece Manual commentato*. GuIt, `http://www.guitex.org/home/images/doc/GuideGuIT/`, 2015.

[5] Claudio Beccari. "La classe TOPtesi". PDF document, 11 2015. In `$TEXMF/doc/latex/toptesi/toptesi.pdf`, `$TEXMF/doc/latex/toptesi/toptesi-it-xetex.pdf`.

[6] Gian Luigi Beccaria. *Il mare in un imbuto*. Giulio Einaudi editore, Torino, 2010.

[7] Francesco Biccari. "Documentation of the LaTeX class sapthesis.cls". PDF document, 3 2012. In `$TEXMF/doc/latex/sapthesis/sapthesis-doc.pdf`.

[8] Robert Bringhurst. *The elements of typographic style*. Hartley & Marks, Vancouver, BC, 2004.

[9] Christophe Caignaert. "A story of *kpfonts*". In: *TGGboat*, vol. 31, no. 3, (2010), pp. 161–174.

[10] CEI 24-1. *Unità di misura e simboli letterali da usare in elettrotecnica*. Comitato Elettrotecnico Italiano, Milano, 1986.

[11] CEI-S-646. *Dizionario della strumentazione nucleare. Primo complemento al fascicolo S-447*. Comitato Elettrotecnico Italiano, Milano, 1983.

[12] Gustavo Cevolani. "Libretti in LaTeX". In: *ArsTEXnica*, vol. 1, no. 2.

[13] Gustavo Cevolani. "Norme tipografiche". In: *ArsTEXnica*, vol. 1, no. 1.

[14] CNR-UNI 10002. *Segni e simboli matematici per le scienze fisiche*

e tecniche. Unificazione Italiana, Milano, 1963.

[15] CNR-UNI 10003. *Sistema internazionale di unità (SI)*. Ente Italiano di Unificazione, Milano, 1984.

[16] Massimiliano Dominici. "Utilizzo di caratteri TrueType con LaTeX – Un esempio pratico: i *FellTypes*". In: *ArsTeXnica*, vol. 2, no. 4, (2007), pp. 88–102.

[17] Massimiliano Dominici. "Una panoramica su Pandoc". In: *ArsTeXnica*, vol. 8, no. 15, (2013), pp. 31–38.

[18] Michael Downes. *Short math guide*. American Mathematical Society, Providence, Rhode Island, 1.09 ed., 2002. In `ftp://ftp.ams.org/pub/tex/doc/amsmath/short-math-guide.pdf`.

[19] Olaf Drümmer, Alexandra Oettler, Dietrich von Seggern. *PDF/A in a Nutshell. Long Term Archiving with PDF*. Callas Software gmbh, 2008. In: `http://www.pdfa.org/doku.php?id=pdfa:en:pdfa_in_a_nutshell`.

[20] Elsevier. "Preparing articles with LaTeX — Instructions to authors for preparing compuscripts". PDF document. In `instraut.pdf`.

[21] Giorgio Fioravanti. *Il manuale del grafico — Guida alla progettazione grafica e all'impaginazione del prodotto editoriale*. Zanichelli, Bologna, 1987.

[22] Peter Flynn. *Formatting information — A beginner's introduction to typesetting with LaTeX*. In `CTAN/tex-archive/info/beginlatex/beginlatex-3.6.pdf`.

[23] Michel Goossens. *The XeTeX Companion – TeX meets OpenType and Unicode*. LaTeX Team, gennaio 2011. In `http://xml.web.cern.ch/XML/lgc2/xetexmain.pdf`.

[24] Enrico Gregorio. "Installare TeX Live 2010 su Ubuntu". In: *ArsTeXnica*, vol. ottobre, no. 10, (2010), pp. 7–13.

[25] Enrico Gregorio. "L'arte esoterica di scrivere in cirillico con LaTeX". In: *ArsTeXnica*, vol. aprile, no. 9, (2010), pp. 57–73.

[26] Enrico Gregorio. "Come comporre un frontespizio e vivere felici". PDF document, 9 2011. In `$TEXMF/doc/latex/frontespizio/frontespizio.pdf`.

[27] Enrico Gregorio. *Introduzione a XeLaTeX*, gennaio 2011. In `http://profs.sci.univr.it/~gregorio/introxelatex.pdf`.

[28] ISO 690. *Documentation – Bibliographical references – Essential and supplementary elements*. International Organization for Standardization, Ginevra, 1982.

[29] ISO Handbook 2. *Units of measurement*. International Organization for Standardization, Ginevra, 1982.

[30] Kluver. "User manual for `kluwer.cls`". In `$TEXMF/doc/latex/kluwer/usrman.dvi`.

[31] Donald E. Knuth. *The TEXbook*. Addison Wesley, Reading, Mass., 16 ed., 1996.

[32] Donald E. Knuth. *Computers & typesetting*. Addison Wesley, Reading, Mass., 2001. Millenium edition.

[33] Helmut Kopka, Patrick W. Daly. *Guide to LATEX*. Addison Wesley, Reading, Mass., 4 ed., 2004.

[34] Steven G. Krantz. *A primer in mathematical writing*. American Mathematical Society, Providence, Rhode Island, 2 ed., 1998.

[35] Leslie Lamport. *A document preparation system — LATEX — User's guide and reference manual*. Addison Wesley, Reading, Mass., 2 ed., 1994.

[36] Jerónimo Leal, Gianluca Pignalberi. *Edizioni Critiche – Guida alla composizione con il proprio computer*. TEXnologie. Edizioni CompoMat, Configni (RI), 2012.

[37] Roberto Lesina. *Il nuovo manuale di stile*. Zanichelli, Bologna, 2 ed., 1994.

[38] André Miede. "A classic thesis style". PDF document, 12 2011. In `$TEXMF/doc/latex/classicthesis/classicthesis.pdf`.

[39] Michael Mitchel, Susan Wightman. *Book typography — A designer's manual*. Libanus Press, Marlborough, Wiltshire UK, 2005.

[40] Frank Mittelbach, Michel Goosens, *et al. The LATEX companion*. Addison Wesley, Reading, Mass., 2 ed., 2004.

[41] Frank Mittelbach, Michel Goosens, *et al. The LATEX graphics companion*. Addison Wesley, Reading, Mass., 2 ed., 2007.

[42] Bice Mortara Garavelli. *Prontuario di punteggiatura*. Editori Laterza, Bari, 11 ed., 2008.

[43] Bice Mortara Garavelli (cur.). *Storia della punteggiatura in Europa*. Editori Laterza, Bari, 2008.

[44] Imprimerie Nationale (cur.). *Règles typogrphiques*. Imprimerie Nationale, Parigi, 2006. ISBN 2-7433-0482-0.

[45] Scott Pakin. "The comprehensive LATEX symbol list". PDF document. In `$TEXMF/doc/latex/comprehensive/symbols-a4.pdf`.

[46] Lorenzo Pantieri. "LATEX per l'impaziente – Un'introduzione all'Arte di scrivere con LATEX". PDF document, settembre 2008. In `http://www.lorenzopantieri.net/LaTeX_files/LaTeXimpaziente.pdf`.

[47] Lorenzo Pantieri. "Tesi Classica". PDF document, 5 2012. In `http://www.lorenzopantieri.net/LaTeX_files/TesiClassica.zip`.

[48] Lorenzo Pantieri. "Tesi Moderna". PDF document, 5 2012. In `http:`

//www.lorenzopantieri.net/LaTeX_files/TesiModerna.zip.

[49] Lorenzo Pantieri, Tommaso Gordini. "L'arte di scrivere con LaTeX". PDF document, 2015. In http://www.lorenzopantieri.net/LaTeX_files/ArteLaTeX.pdf.

[50] Will Robertson. "The XeTeX reference guide". PDF document, gennaio 2011. In $TEXMF/doc/xetex/xetexref/XeTeX-reference.pdf.

[51] Francesca Serafini. *Questo è il punto – Istruzioni per l'uso della punteggiatura*. Editori Laterza, Bari, 2012.

[52] Fabrizio Serra. *Regole editoriali, tipografiche & redazionali*. Istituti Editoriali e Poligrafici Internazionali, Pisa – Roma, 2004. Prefazione di Martino Mardersteig, postfazione di Alessandro Olschki, appendice di Jan Tschichold.

[53] N. Setzer. "The cool package". PDF document. In $TEXMF/doc/latex/cool/cool.pdf.

[54] Societa Italiana di Fisica e Il Nuovo Cimento, Bologna. *Guida per gli autori*, 1988.

[55] Ellen Swanson, Arlene O'Sean, Antoinette Schleyer. *Mathematics into type*. American Mathematical Society, Providence, Rhode Island, 2 ed., 1999.

[56] Till Tantau. "Ti*k*Z and pgf — Manual for version 1.10". PDF document. In texlive/2012/texmf-dist/doc/generic/pdf/version-for-pdftex/en/pgfmanual.pdf.

[57] A. Thompson, B. N. Taylor. "The NIST Guide for the use of the International System of Units", 2008. In http://physics.nist.gov/cuu/pdf/sp811.pdf.

[58] Edward R. Tufte. *The visual display of quantitative information*. Grapphics Press, Cheshire, Connecticut, 2 ed., 2007.

[59] UNI 2949. *Diagrammi e cartogrammi, Regole generali per l'elaborazione*. Ente Italiano di Unificazione, Milano, 1982.

[60] UNI 6015. *Segnaccento obbligatorio nell'ortografia della lingua italiana*. Ente Italiano di Unificazione, Milano, 1967.

[61] UNI 6017. *Descrizioni e riferimenti bibliografici*. Ente Italiano di Unificazione, Milano, 1967.

[62] UNI 6461. *Divisione delle parole in fin di linea*. Ente Italiano di Unificazione, Milano, 1969.

[63] UNI 7090. *Metodo di scrittura numerica delle date*. Ente Italiano di Unificazione, Milano, 1973.

[64] UNI-ISO 5966. *Presentazione dei rapporti scientifici e tecnici*. Ente Italiano di Unificazione, Milano, 1989.

[65] Ivan Valbusa. "User's guide to suftesi". PDF document, 2015. In $TEXMF/doc/latex/suftesi/suftesi.pdf.

[66] Herbert Voss. "Math mode". PDF document, 2010. In $TEXMF/doc/latex/mathmode/.

[67] R.C. Weast, M.J. Astle, W.H. Beyer (cur.). *CRC handbook of chemistry and physics*, CRC Press, Boca Raton, Florida, cap. "Symbols, units and nomenclature in physics", pp. F259–F293. 65 ed., 1984.

[68] Peter Wilson. "The memoir class for configurable typesetting — User guide". PDF document, 2004. In `\protect\T1\textdollarTEXMF/doc/latex/memoir/memman.pdf`.

[69] Peter Wilson. *A few notes on book design*. The Herries Press, Normandy Park, WA, 2009.

Indice analitico

Simboli

!, 607, 676
\!, 285, 654
\", 635
", 290, 400
"<, 630
">, 630
#, 99
$, 99
%, 99
&, 99
\', 635, 681, 682
', 630
'', 630
\(, 263, 283, 651, 655, 669
\), 263, 283, 651, 655, 669
\+, 681
\,, 293, 630, 654
\-, 27, 334, 401, 559, 561, 566, 580, 681
-, 630
--, 630
---, 630
\., 635
\:, 290, 654
\;, 654
\<, 681
<, 169, 672
≪, 630
\=, 635, 681, 682
=, 672
\>, 681
>, 169, 672
≫, 630
@, 236, 400, 405, 416
\@, 630
@-espressione, 156, 683
\@dottedtocline, 407, 416
\@evenhead, 466
\@glossaryfile, 246
@gobble, 414
\@ifdefinable, 128
\@ifundefined, 128
\@indexfile, 246
\@oddhead, 466
\@Roman, 406, 409
\@roman, 405, 408
\@sptoken, 411
\@tfor, 412
\[, 253, 264, 290, 651
\~, 635
\\, 143, 144, 157, 159, 164, 281, 401, 602, 627–629, 643, 644, 681, 683, 684, 690
*, 281, 629, 690
\^, 635
\], 253, 264, 290, 651
^, 99
\|, 257, 258
_, 99
\‘, 635, 681, 682
‘, 414, 630
‘‘, 630
{, 99
}, 99

~, 400, 602, 630, 644
␣, 630

A

\a', 682
\a=, 682
\a‘, 682
\AA, 635
\aa, 635
\abovedisplayshortskip, 652
\abovedisplayskip, 652
\accent, 559, 589
\addcontentsline, 422, 637
\addconttentsline, 637
\addto, 404, 560
\addtocounter, 627, 659
\addtolength, 663
\addvspace, 673
\advance, 664
\AE, 635
\ae, 635
\AfterEndPreamble, 617
\afterpage, 167, 168, 176, 427, 428, 607
\aleph, 258
\allowdisplaybreak, 281
\allowdisplaybreaks, 280, 281
\Alph, 660
\alph, 660
\alpha, 253, 254
\amalg, 255
\and, 644, 669
\angle, 258
\ap, 266, 652
\appendix, 108, 118
\approx, 255
\arabic, 659, 661
\arc, 182
\arc*, 182
\arccos, 256
\arcsin, 256
\arctan, 256
\arg, 256
\arraybackslash, 169
\arraycolsep, 684
\arrayrulewidth, 684
\arraystretch, 160, 684
\Arrowvert, 257
\arrowvert, 257
\ast, 255
\asymp, 255
\AtBeginDocument, 411, 670
\Author, 494
\author, 329, 469, 629, 644
\autore, 469

B

\b, 635
\backmatter, 108, 112, 118, 466, 636
\backslash, 257, 258
\baselineskip, 423, 455, 632, 691
\beamertemplatetransparentcovereddynamic, 329
\begin, 105, 132–134, 136, 137, 139, 142, 154, 156, 175, 179, 264, 290, 418, 472, 568, 614, 629, 645, 649–651, 657, 674, 681, 682, 686–688, 693, 695, 696
\begingroup, 142
\belowdisplayshortskip, 652
\belowdisplayskip, 652
\beta, 253, 254
Beta Code, 300
\bfseries, 171, 349, 358, 705
\bgroup, 142, 419, 697
\bibindent, 639
\bibitem, 139–142, 628, 686
\bibliography, 232, 238, 685
\bibliographystyle, 232
\bigbreak, 692
\bigcap, 256
\bigcirc, 255
\bigcup, 256
\Biggl, 262
\biggl, 262
\Biggr, 262

\biggr, 262
\Bigl, 262
\bigl, 262, 277
\bigodot, 256
\bigoplus, 256
\bigotimes, 256
\Bigr, 262
\bigr, 262, 277
\bigskip, 673
\bigskipamount, 692
\bigsqcup, 256
\bigtriangledown, 255
\bigtriangleup, 255
\biguplus, 256
\bigvee, 256
\bigwedge, 256
\binom, 282
\bm, 286, 363
\boldmath, 286, 361, 655
\boldsymbol, 286, 363
\boolean, 668
\bot, 258
\botfigrule, 176, 674
\botmark, 465, 467
\bottomfraction, 677, 679
\bottomrule, 161
\bowtie, 255
\Box, 258
\box, 420, 698, 699
\boxed, 287
\bracevert, 257
\break, 334
\bullet, 255
\buonfinepagina, 609

C

\c, 635
\cap, 255
\capolettera, 402
\capoverso, 402
\caption, 146, 155, 168, 172, 635, 636, 676, 677, 699
\cbezier, 181, 182
\cdot, 255
\cdots, 653
\centering, 142–145, 169, 629, 683
\cfrac, 282
\chapter, 111, 112, 423, 465, 466, 520, 635, 636
\chapter*, 96, 637
\chaptermark, 466
\chaptername, 635, 636
\char, 401, 589
\chi, 254
chiave
  angle, 227
  bb, 225
  clip, 227
  height, 226
  keepaspectratio, 226
  trim, 227
  viewport, 226
  width, 226
\circ, 255
\circle, 180
\circle*, 180
\cite, 140–142, 233, 312, 503, 686
\cleaders, 424, 426
\cleardoublepage, 669, 676, 691
\clearpage, 176, 601, 605, 607, 676, 687, 691
\clerapage, 607
\cline, 159, 160, 289, 684
\closeout, 247
\clubpenalty, 609
\clubsuit, 258
codice di allineamento
  B, 702
  b, 157, 682, 696, 702
  c, 157, 682, 696
  l, 693, 702
  r, 693, 702
  s, 693
  t, 157, 682, 696, 702
colophon, 469
\color, 704
\colorbox, 704
\columnsep, 632
\columnseprule, 632

\columnwidth, 226, 632, 664
\cong, 255
contatore
  bottomnumber, 677–679
  dbltopnumber, 678, 679
  footnote, 633
  MaxMatrixCols, 287
  secnumdepth, 635–637
  table, 172
  tocdepth, 635–637
  topnumber, 677–679
  totalnumber, 677–679
\coprod, 256
\copyright, 635
\cos, 256
\cosh, 256
\cot, 256
\coth, 256
\cs, 401
\csc, 256
\csname, 434, 649
\cspreto, 318
\cup, 255

D

\D, 161
\d, 635
\dag, 635
\dagger, 255
\dashbox, 180, 181
\dashv, 255
\date, 329, 469, 629, 644
\dbinom, 282
\dblfigrule, 176, 674
\dblfloatpagefraction, 678, 679
\dblfloatsep, 678, 679
\dbltextfloatsep, 678, 679
\dbltopfraction, 678, 679
\ddag, 635
\ddagger, 255
\ddots, 653
\DeclareGraphicsRule, 212, 220
\DeclareMathOperator, 259, 281, 285
\DeclareMathOperator*, 259
\DeclareMathSizes, 705
\DeclareMathSymbol, 411, 617
\DeclareOperatorName, 259
\DeclareRobustCommand, 417
\def, 128, 161
\definecolor, 703
\deg, 256
\Delta, 253, 254
\delta, 254
\depth, 693
descrittore di colonna
  @, 156, 683
  |, 156, 683
  b, 159, 164, 628
  C, 171
  c, 156, 171, 683
  D, 168
  E, 171
  L, 171
  l, 156, 171, 683
  m, 159, 164, 628
  p, 156, 159, 164, 628, 683, 684
  r, 156, 171, 683
  S, 164
  s, 164
  X, 164
\det, 256
\dfrac, 282, 283
\Diamond, 258
\diamond, 255
\diamondsuit, 258
\diff, 285
\dim, 256
\dimexpr, 663, 665
\displaybreak, 280, 281
\displaystyle, 172, 252, 492, 656
\div, 255
\divide, 664, 666
\do, 412
\documentclass, 93, 96, 104, 122, 167, 352, 629, 638, 687
\doteq, 255
\dotfill, 424, 426
\dots, 653, 654

\doublerulesep, 684
\Downarrow, 255, 257
\downarrow, 255, 257
\dp, 700

E

\edef, 128
\egroup, 142, 419
\ell, 258
\else, 411, 412, 616
\em, 359, 560
\emph, 75, 172, 359, 560, 631
\emptyset, 258
\end, 95, 105, 128, 133, 134, 136, 137, 139, 142, 154, 156, 175, 179, 264, 418, 472, 568, 614, 629, 645, 650, 651, 657, 674, 676, 681, 682, 686–688, 693, 695, 696
\endcsname, 434, 649
\endfirsthead, 172
\endfoot, 172
\endgroup, 142
\endhead, 172
\endinput, 126
\endlastfoot, 172
\endlist, 649
\endmulticols, 423
\enlargethispage, 457, 608, 633, 691, 692
\enlargethispage*, 633, 691
\enspace, 171, 290
\ensuremath, 651
\epsilon, 254
\eqref, 265
\equal, 667
\equiv, 255
\ergoup, 697
\errorstopmode, 620
\eta, 254
\etichettadescrizione, 650
\evensidemargin, 643
\exists, 258, 272
\exp, 256
\expandafter, 412, 414, 621, 649
\extracolsep, 162, 173, 683

F

\f@baselineskip, 405
\fbox, 430, 693, 695, 696, 704
\fboxrule, 419, 693, 704
\fboxsep, 419, 693, 704
\fcolorbox, 704
\fi, 411, 412, 616
\figurename, 635, 636
\filbreak, 609, 692
file
  di formato, 79
  di inizializzazione, 81
\fill, 162, 662, 683
\finalhyphendemerits, 334, 601
\firstmark, 465, 467
\flat, 258
\floatpagefraction, 677–679
\floatsep, 678, 679
\fnsymbol, 660
\fontencoding, 350
\fontseries, 390
\fontshape, 390
\fontsize, 406
\footnote, 145, 146, 503, 504, 558, 628, 633
\footnotemark, 146, 147, 633
\footnoterule, 633, 674
\footnotesep, 633
\footnotesize, 165, 166, 353, 446, 705
\footnotetext, 146, 147, 628, 633
\footskip, 459, 460
\forall, 258
\frac, 282, 653
\frakfamily, 306
\framebox, 180, 181, 420, 430, 693
\frenchspacing, 477, 630, 631
\frontmatter, 108, 112, 118, 466, 636, 637
\frown, 255

\fussy, 690

G

\Gamma, 253, 254
\gamma, 253, 254
\gcd, 256
\gdef, 128, 411
\ge, 255
\geq, 255
\gets, 255
\gg, 255
\glossary, 244, 245, 638, 685, 689
\glossaryentry, 685
\glueexpr, 663
\goodbreak, 609, 692
\goodpagebreak, 609
\gothfamily, 306
\graphicspath, 222–224
\greekfont, 315
\GuIT, 94

H

\H, 635
\halign, 173
\hangafter, 403
\hangindent, 403
\hat, 99
\hbadness, 565
\hbar, 258
\hbox, 406, 431, 695, 697, 698
\hbox to, 698
\header, 459
\headsep, 459, 460
\heartsuit, 258
\height, 693
\hfill, 425, 468, 673
\hfuzz, 565
\hline, 159, 684
\hom, 256
\hookleftarrow, 255
\hookrightarrow, 255
\hrule, 172, 425, 433
\hrulefill, 424, 425
\hsize, 419
\hskip, 424, 559
\hspace, 673
\hspace*, 673
\hss, 426, 649, 698
\ht, 700
\Huge, 353, 700, 705
\huge, 353, 403, 705
\hyphenation, 26, 27, 562, 564, 568, 627
\hz, 560

I

\i, 505, 587, 635
\idotsint, 284
\if, 412
\ifbool, 671
\ifcat, 414
\ifcsdef, 671
\ifcsempty, 671
\ifcsequal, 671
\ifcsname, 414
\ifcsprefix, 671
\ifcsstring, 672
\ifcsundef, 671
\ifdef, 671
\ifdefempty, 671
\ifdefequal, 671
\ifdefprefix, 671
\ifdefstring, 672
\iff, 255
\ifFamily, 313
\iflanguage, 415
\iflettere, 409
\ifmmode, 671
\ifnum, 413, 616, 661
\ifnumcomp, 672
\ifnumequal, 672
\ifnumgreater, 672
\ifnumless, 672
\ifnumodd, 672
\ifodd, 428, 661, 671, 672
\ifthen, 673
\ifthenelse, 667, 669

\ifundef, 671
\ifx, 411–413
\iiiint, 284
\iiint, 284
\iint, 284
\Im, 258
\imath, 258, 262
\immediate, 247
\in, 255, 272
\include, 101–106, 614, 685, 687, 691
\includegraphics, 101, 191, 220–223, 225, 227, 330, 431, 433, 511, 679, 703
\includegraphics*, 225, 227
\includeonly, 101–103, 105, 106, 108, 685, 687
\includesvg, 222
\indent, 144, 631
\index, 241, 242, 245, 249, 250, 401, 638, 641, 685, 688, 689
\indexentry, 242, 685
\indexname, 422
\indexprologue, 250
\indexspace, 421
\indici, 118
\inf, 256
\infty, 258, 414
\input, 101–104, 106, 168, 243, 247, 687, 688
\insertpenalties, 606
\institution, 329
\int, 256, 284
\IntelligentComma, 132
interprete TEX, 83
\intertext, 282, 283, 631, 704
\intextsep, 678, 679
\iota, 254
\isodd, 668
\isundefined, 668
\it, 357
\item, 136, 137, 139, 330, 423, 559, 627, 645, 646
\itemindent, 646
\itemsep, 646, 649
\itshape, 171, 349, 358, 705

J

\j, 635
\jmath, 258, 262
\jobname, 246–248, 685
\Join, 255
\jot, 652
\justify, 142

K

\kappa, 254
\ker, 256
\Keywords, 494
\kill, 681

L

\L, 635
\l, 635
\l@chapter, 408
\l@figure, 416
\l@table, 416
\label, 138, 141, 146, 149, 155, 264, 265, 273, 280, 633, 647, 652, 657, 659, 677, 686
\labelsep, 646
\labelwidth, 646
\Lambda, 254
\lambda, 254
\langle, 257
\LARGE, 353, 705
\Large, 353, 705
\large, 353, 705
larghezza del font, 336
\LaTeX, 631
\LaTeXe, 631
\lceil, 257
\ldots, 653
\le, 255
leader, 424
\leaders, 424, 426
\leadsto, 255

\leavevmode, 425
\left, 261, 262, 277, 361
\Leftarrow, 255
\leftarrow, 255
\lefteqn, 652
\leftharpoondown, 255
\leftharpoonup, 255
\lefthypenmin, 568
\lefthyphenmin, 568
\leftmargin, 646, 649
\leftmargini, 646, 649
\leftmarginii, 646
\leftmarginiii, 646
\leftmarginiv, 646
\leftmark, 465–467
\Leftrightarrow, 255
\leftrightarrow, 255
legature e segni diacritici, 337
\lemma, 467, 468
\lengthtest, 667
\leq, 255
\let, 414, 649
\let@token, 411
\lfloor, 257
\lg, 256
\lgroup, 257
\lhd, 255
\lim, 256
\liminf, 256
\limsup, 256
\Line, 182
\line, 180
linea guida, 424
\linebreak, 334, 627, 690
\linepenalty, 603
\lineskip, 605, 632
\lineskiplimit, 632
\linespread, 356, 357, 632
\linethickness, 181
\linewidth, 166, 226, 419, 632, 696
\list, 649
\listfigurename, 404
\listfiles, 688
\listoffigures, 177, 404, 635
\listoftables, 177, 635
\listparindent, 646
\ll, 255
\llap, 403, 649
\lmoustache, 257
\ln, 256
\log, 256
\logo, 329
\long, 656, 671
\Longleftarrow, 255
\longleftarrow, 255
\Longleftrightarrow, 255
\longleftrightarrow, 255
\longmapsto, 255
\Longrightarrow, 255
\longrightarrow, 255
\looseness, 457, 608, 692
\LTleft, 172
\LTright, 172

M

\m@thcomma, 411
\MacTeX, 401
\maimatter, 636
\mainmatter, 108, 112, 118, 466, 636
\makeatletter, 405
\makeatother, 247
\makebox, 166, 180, 181, 185, 430, 693, 695, 698
\makeglossary, 244, 638, 685, 689
\makeindex, 241, 249, 638, 685, 688
\makelabel, 646
\MakeLowercase, 308
\maketaletter, 247
\maketitle, 469, 644, 660
\MakeUppercase, 308, 422, 466
\mapsto, 255, 272, 491, 492
\marginpar, 147, 148, 633, 679
\marginparpush, 634, 680
\marginparsep, 634, 680
\marginparwidth, 634, 680
\markboth, 423, 465–467, 637, 642
\markright, 423, 465–467, 637, 642
\masthindent, 640

\mathbb, 271
\mathbf, 286, 655
\mathcal, 290, 361, 655
\mathchoice, 491, 492
\mathindent, 640, 652
\mathit, 254, 364, 655
\mathnormal, 492
\mathop, 280, 285
\mathord, 411
\mathpunct, 411
\mathr, 263
\mathrel, 492
\mathrm, 263, 655, 704
\mathsf, 364, 655
\mathtt, 364, 655
\max, 256
\maxdimen, 620, 665
\mbox, 263, 430, 631, 693, 694, 697, 698
\mdseries, 349, 358, 705
\medbreak, 692
\medskip, 172, 673
\medskipamount, 692
METAPOST, 89
\mho, 258
\mid, 255
\middle, 361
\midrule, 161
\min, 256
\mkern, 654
\models, 255
\mp, 255
\mskip, 654
\mu, 254
\muexpr, 663
\multicolumn, 158, 684
\multiply, 664, 666
\multiput, 179, 181

N

\nabla, 258
\natural, 258
\ne, 255
\nearrow, 255
\NeedsTeXFormat, 126
\neg, 258
\neq, 255, 491
\newbool, 671
\newboolean, 668
\newbox, 698
\newcolumntype, 170, 171, 683
\newcommand, 127, 128, 280, 285, 400, 402, 404, 416–418, 424, 656, 690
\newcommand*, 400, 467
\newcount, 672
\newcounter, 627, 658, 672
\newenvironment, 418, 420, 421, 657, 698
\newfloat, 472
\newif, 668, 671
\newlength, 627, 663
\newline, 157, 629, 690
\newpage, 457, 609, 669, 691
\newsavebox, 430, 627, 694, 695, 698
\newtheorem, 117, 627, 657
\nexists, 272
\ni, 255
\noalign, 172
\nobreak, 334, 560
\nocite, 233, 238, 239
\noexpand, 414
\nofiles, 685
\noindent, 144, 419, 631
\NoIntelligentComma, 132
\nolinebreak, 627, 690
\nonfrenchspacing, 631
\nonumber, 652
\nopagebreak, 627, 691
\normalbaselineskip, 455
\normalfont, 357, 705
\normalmarginpar, 634, 679, 680
\normalsize, 353, 705
\not, 255, 272, 491, 669
\notag, 274
\notbool, 671
\notin, 272
\nu, 254

\null, 193, 468
\numberline, 638
numerazione
  Alph, 642, 660
  alph, 642, 660
  arabic, 642, 659
  fnsymbol, 642, 660
  Roman, 642, 660
  roman, 642, 659
\numexpr, 663, 672
\nwarrow, 255

O

\O, 635
\o, 635
occhiello, 469
\oddsidemargin, 643
\odot, 255
\OE, 635
\oe, 635
\ohm, 266, 417
\oint, 256
\oldstylenums, 364
\Omega, 254, 266
\omega, 254
\ominus, 255
\onecolumn, 643
\oplus, 255
opzione
  10pt, 446, 456, 638, 648
  11pt, 352, 356, 446, 639, 705
  12pt, 352, 446, 639
  a-1b, 493
  a4paper, 639
  a5paper, 639
  ansinew, 65, 95, 579
  applemac, 95, 579, 580, 585, 591
  b5paper, 639
  babelshorthands, 27
  backend, 240
  backref, 240
  charter, 365
  columns, 249
  config, 339
  cp1250, 580
  cp1252, 579
  cp437, 580
  draft, 639
  dvipdfm, 510
  enable-write18, 217
  EU1, 318
  executivepaper, 639
  final, 639
  fleqn, 640, 652
  garamond, 365
  greek, 299, 300
  hyperref, 240
  ibycus, 300
  intoc, 249
  italian, 122, 132, 133, 267, 290, 328, 410, 505, 652, 653
  landscape, 322, 639
  latin, 307
  latin1, 68, 95, 509, 510, 579, 580, 591, 593, 597, 600
  latin9, 95, 510, 579, 580
  legalpaper, 639
  leqno, 640
  letterpaper, 639
  LGR, 311, 313, 571
  LMR, 313
  ltxarrows, 178
  LY1, 377
  main, 367
  natbib, 240
  nome, 249
  notitlepage, 639
  oldstyle, 305
  onecolumn, 639
  oneside, 639
  openany, 639
  openbib, 639
  openright, 639
  options, 249
  origin, 702
  original, 178
  OT1, 95, 305, 589, 590
  OT2, 367, 368
  pdfa, 493

polutoniko, 301
pstarrows, 178
shell-escape, 217
square, 240
style, 239
T1, 94, 95, 305, 313, 318, 328, 367, 377, 388, 389, 570, 572, 581, 585, 588–590, 598
T2, 377
T2A, 367, 369
T2B, 367
T2C, 367
title, 249
titlepage, 639, 643
TS1, 377, 581
twocolumn, 639
twoside, 639, 642
utf8, 59, 60, 66, 68, 94, 95, 303, 309, 328, 369, 503, 571, 572, 580, 581, 590–593, 597, 598, 600
utopia, 365
veryoldstyle, 305
x-1a, 493
X2, 367–369
\or, 669
\Org, 494
\oslash, 255
\otimes, 255
\oval, 180, 181
\overbrace, 262
\overline, 654

P

\P, 635
\pagebreak, 281, 601, 627, 691
\pagecolor, 627, 704
\pagenumbering, 627, 642
\pageref, 139, 141, 155, 265, 652, 686
\pagestyle, 641
\paperheight, 665
\paperwidth, 665
\par, 143, 144, 402, 419, 423, 632, 656
\paragraph, 293, 635, 636
\parallel, 255
\parbox, 193, 430, 433, 656, 683, 696, 698
\parindent, 632
\parsep, 646, 648
\parskip, 605, 632
\part, 112, 635, 636
\partial, 258
\partname, 635, 636
\partopsep, 646
\pdfoutput, 83, 337
\ped, 266, 653
\perp, 255
\pgfdeclareimage, 329
\pgfuseimage, 329, 330
\Phi, 254
\phi, 254
\Pi, 254
\pi, 254
piedino, 17
\pm, 255
\pmb, 286
\polygon, 182, 183
\polygon*, 182, 183
\polyline, 182, 183
\poptabs, 682
posizione degli oggetti flottanti
  !, 154, 176, 674
  b, 154, 176, 674
  H, 176
  h, 154, 176, 674, 675
  p, 154, 176, 674, 677
  t, 154, 176, 674, 675
\pounds, 635
\Pr, 256
\prec, 255
\preceq, 255
\pretolerance, 335, 425, 603
\prime, 258
\printglossary, 244, 248
\printindex, 243, 244, 249, 250, 476, 688

\prod, 256
\propto, 255
\protect, 417, 628, 638, 671
protrusione dei segni grafici, 336
protrusione della punteggiatura, 336
\providebool, 671
\provideboolean, 668
\providecommand, 416, 420, 656
\providecommand*, 416
\ProvidesClass, 127
\ProvidesPackage, 126
\ProvideTextCommandDefault, 586
\Psi, 254
\psi, 254
\pushtabs, 681
\put, 179–182, 191

Q

\qbezier, 181, 182
\qquad, 276, 293, 346, 654
\quad, 276, 293, 346, 654

R

\r, 635
\raggedleft, 142, 148, 157, 169, 629, 680, 683
\raggedright, 142, 148, 157, 169, 629, 683
\raisebox, 403, 697
\rangle, 257
\rBrace, 192, 193
\rceil, 257
\Re, 258, 259
\ref, 139, 141, 149, 155, 265, 633, 647, 652, 657, 659, 686
\reflectbox, 703
\refstepcounter, 659, 686
refuso, 21
\relax, 128, 405, 616, 663, 664, 672, 674
\renewcommand, 127, 128, 160, 404, 405, 420, 634, 656, 678, 684
\renewcommand*, 404
\renewenvironment, 420, 421, 657
\RequirePackage, 428
\resizebox, 166, 702
\reversemarginpar, 147, 634, 679, 680
\rfloor, 257
\rgroup, 257
\rhd, 255
\rho, 254
\right, 261, 262, 277, 361
\Rightarrow, 255
\rightarrow, 255
\rightharpoondown, 255
\rightharpoonup, 255
\righthyphenmin, 568
\rightleftharpoons, 255
\rightmargin, 646
\rightmark, 465–467
risposta all'errore
  e, 109
  h, 109
  i, 109
  q, 109
  s, 109
  x, 109
\rlap, 492
\rm, 357
\rmfamily, 348, 357, 358, 705
\rmoustache, 257
\Roman, 660
\roman, 405, 659
\romannumeral, 409
\rotatebox, 702
\rule, 160, 161, 433, 492, 697

S

\S, 635
\savebox, 430, 695
\sbox, 430, 694, 698
\scalebox, 702
\scriptscriptstyle, 252, 492, 656
\scriptsize, 263, 353, 705

\scriptstyle, 252, 492, 656
\scshape, 349, 358, 705
\searrow, 255
\sec, 256
\section, 75, 112, 423, 465, 466, 520, 635, 636
\section*, 637
\selectfont, 350, 390, 406, 632
\selectlanguge, 404
\setactivedoublequote, 132, 290
\setbeamercolor, 329
\setboolean, 668
\setbox, 419, 695, 698, 699
\setcounter, 287, 627, 637, 658, 659, 678
\setISOcompliance, 132
\setlength, 178, 640, 663, 678
\setminus, 255
\settodepth, 663, 700
\settoheight, 663, 700
\settowidth, 663, 700
\sf@size, 405
\sfdefault, 318
\sffamily, 348, 358, 705
\sharp, 258
shell editor, 62
\shipoutAnswer, 150
\shorthandoff, 290, 291
\shorthandon, 290, 291
\shortstack, 629
\showboxbreadth, 619, 620
\showboxdepth, 619, 620
\showhyphens, 558, 564
\SI, 267
\Sigma, 254
\sigma, 254
\sim, 255
\simeq, 255
\simulatedCS, 410
\simulatedSC, 406, 408
\sin, 256
\sinh, 256
\sloppy, 690
\slshape, 349, 358, 705
\small, 165, 353, 446, 705
\smallbreak, 691
\smallskip, 172, 673
\smallskipamount, 691
\smile, 255
\spadesuit, 258
spazieggiatura, 336
\special, 79
\sqcap, 255
\sqcup, 255
\sqrt, 256, 653
\sqsubset, 255
\sqsubseteq, 255
\sqsupset, 255
\sqsupseteq, 255
\ss, 635
\stackrel, 654
\star, 255
\stepcounter, 659
stile della pagina
  empty, 423, 642
  headings, 423, 466, 642
  myheadings, 423, 466, 642
  plain, 423, 642
\stretch, 662
\stretchandshrink, 173
\string, 414
\strut, 159, 160, 629, 633
\subparagraph, 111, 635, 636
\subsection, 465, 520, 635, 636
\subset, 255
\subseteq, 255
\substack, 284
\subsubsection, 520, 635, 636
\subtitle, 329
\succ, 255
\succeq, 255
\sum, 256
\sup, 256
\supaginadispari, 428
\supaginapari, 428
\suppressfloats, 627, 677
\supset, 255
\supseteq, 255
\surd, 258
\swabfamily, 306

\swarrow, 255
\symbol, 706

T

\t, 635
\T1/lmr/m/n/10, 390
\tabcolsep, 164, 165, 684
\tablename, 635, 636
\tableofcontents, 635
\tabucline, 289
\tabularimage, 433, 434
\tabularnewline, 159, 164
\tabulinestyle, 289
\tag, 265, 274
\tan, 256
\tanh, 256
\tau, 254
testatina, 17
\testlettere, 409, 410
\TeX, 401, 631
\text, 263, 279, 283, 631, 704
\textasciicircum, 100
\textasciitilde, 100
\textbackslash, 100, 401
\textbf, 286, 349, 358, 705
\textcelsius, 504
\textcent, 586
\textcolor, 704
\textcurrency, 586
\textcyrillic, 368
\textdegree, 504, 586
\textdollar, 100
\texteuro, 171, 377
\textfloatsep, 678, 679
\textfraction, 677, 679
\textfrak, 306
\textgoth, 306
\textheight, 226, 455, 643, 665
\textit, 349, 655, 705
\textmd, 349, 705
\textmho, 366
\textnormal, 705
\textnumero, 34
\textohm, 266
\textormath, 417, 651
\textrm, 348, 357, 358, 631, 705
\textsc, 349, 405, 406, 705
\textsf, 348, 358, 705
\textsl, 349, 358, 705
\textstyle, 190, 252, 492, 656
\textsuperscript, 653
\textswab, 306
\texttt, 348, 401, 705
\textunderscore, 100
\textup, 349, 705
\textvisiblespace, 578
\textwidth, 226, 446, 632, 643, 664, 665
\tfrac, 282
\thanks, 628, 644, 660
\the, 657, 658, 661, 701
\thepage, 406
\Theta, 254
\theta, 254
\thetable, 172
\thicklines, 181
\thinlines, 181
\thinspace, 290
\thispagestyle, 423, 627, 641
thumbnail, 78
\times, 255
\tiny, 353, 705
\Title, 494
\title, 329, 469, 629, 643
\titlepage, 330
\titolo, 469
\to, 255
\today, 631
\tolerance, 335, 425, 568, 603
\top, 258
\topcaption, 168
\topfigrule, 176, 674
\topfraction, 677, 679
\topmargin, 643
\topmark, 465, 467
\toprule, 161
\topsep, 646, 649
\topskip, 455, 456
\totalheight, 693

\traceoff, 619, 620, 625
\traceon, 619, 620, 625
\tracingassigns, 618
\tracingcommands, 618, 619
\tracinglostchars, 618
\tracingmacros, 618, 619
\tracingonline, 618
\tracingoutput, 618
\tracingpages, 618, 622
\tracingparagraphs, 618, 622
\tracingrestores, 619
\tracingstats, 618
\triangle, 258
\ttfamily, 348, 358, 705
\twocolumn, 421, 627, 643
\twocolumns, 675
\typein, 628, 689
\typeout, 628, 689, 690

U

\U, 161, 257
\u, 257, 635
\uimm, 259, 280
\uishape, 357, 358
\unboldmath, 286, 361, 655
\underbrace, 262
\underline, 654
\unhbox, 699
\unhcopy, 699
\unit, 132, 265–267
unità di misura
  bp, 662
  cc, 662
  cm, 662
  dd, 662
  em, 662
  ex, 662
  in, 662
  mm, 662
  pc, 662
  pt, 662
  sp, 662
unità di misura, 10
  big point, 10, 662
  cicero, 12
  pica, 12
  punto didot, 12
  punto PostScript, 10
  punto tipografico anglosassone, 10
\unita, 267
\unitlength, 178, 179, 193
\unitre, 381
\unlhd, 255
\unrhd, 255
\unvbox, 420, 699
\unvcopy, 699
\Uparrow, 255, 257
\uparrow, 255, 257
\uplus, 255
\upshape, 349, 358, 705
\Upsilon, 254
\upsilon, 254
\usapackage, 428
\usebox, 430, 695, 699
\usecounter, 647
\useoutertheme, 329
\usepackage, 94, 121, 126, 141, 178, 249, 405, 640, 687
\usetheme, 328, 331

V

\V, 161, 257
\v, 635
\value, 428, 659, 661, 664, 667
\varDelta, 257
\varepsilon, 254
\varGamma, 257
\varphi, 254
\varpi, 254
\varrho, 254
\varsigma, 254
\varTheta, 257
\vartheta, 254
\vbadness, 566
\vbox, 419, 431, 698
\vbox to, 698
\vcenter, 431, 698

\vcenter to, 698
\vdash, 255
\vdots, 653
\vector, 180
\vee, 255
\verb, 650
\verb*, 650
\vert, 257
\vfill, 673
\vfuzz, 566
\virgola, 411, 412, 414, 617
virgola intelligente, 132
\virgoladecimale, 411, 412, 414
\vline, 156, 683
\vspace, 144, 402, 423, 673
\vspace*, 144, 673
\vsplit, 699
\vss, 698
\vtop, 431, 698, 699
\vtop to, 698

W

\wd, 700
\wedge, 255
\whiledo, 471, 669
\widowpenalty, 609
\width, 472, 693
\wp, 258
\wr, 255
\write18, 246, 247

X

\xdef, 128
\Xi, 254
\xi, 254
\xleaders, 424, 426
\xleftarrow, 284
\xrightarrow, 284

Z

\z@, 560
\zeta, 254

Indice degli ambienti

A

abstract, 112, 644
align, 277, 278, 280, 293
alignat, 279
aligned, 273–275, 280
alignedat, 280
alltt, 640, 650
amsmath, 276
Answer, 149
array, 159, 164, 289, 628, 629, 682

B

Bmatrix, 287
bmatrix, 287

C

cases, 261, 283
CD, 290
center, 142, 144, 145
checkhyphens, 558

D

description, 139, 645, 649
descrizione, 650
displaymath, 253, 264, 265, 274, 651
document, 96, 167, 501

E

enumerate, 136, 647, 649
eqnarray, 273, 274, 629, 651, 652
*eqnarray**, 273, 651
equation, 264, 274, 277, 651, 652
*equation**, 264, 265, 274
Exercise, 149

F

figure, 144, 175, 177, 178, 221, 471, 677
filecontents, 629, 688
flalign, 278
flushleft, 142, 508
flushright, 142
frame, 324, 330

G

gather, 277
gathered, 280
glossary, 638

I

index, 638
itemize, 137

L

letter, 628
list, 645–649
longtable, 172, 173
longtabu, 164

lrbox, 695, 696

M

math, 651
matrix, 287
medaglione, 418, 419
minibox, 560
minipage, 146, 430, 434, 560, 633, 683, 695–697
multicols, 339, 421, 423
multline, 275, 276, 280

P

picture, 177–179, 181, 183, 185, 186, 189, 191, 193, 322, 331, 400, 443, 641, 701
pmatrix, 287

Q

quotation, 134
quote, 133, 134

R

riquadro, 418, 695, 696

S

sidewaystable, 167
sintassi, 418, 419
sloppypar, 568
split, 273–275, 280
subequations, 274, 279, 280

T

tabbing, 629, 681
table, 144, 154, 160, 161, 175–177, 193, 471, 677
tabu, 164, 169, 174, 289
tabular, 146, 155, 156, 164, 177, 191, 628, 629, 682, 683
*tabular**, 156, 164, 682, 683
tabularx, 164
thebibiography, 237
thebibliography, 139, 140, 231, 237, 686
theglossary, 245
theindex, 243, 245, 420, 423, 688
tikzpicture, 186, 187, 189, 701
titlepage, 469, 644, 660

V

verbatim, 135, 650
verse, 134
Vmatrix, 287
vmatrix, 287

W

wrapfigure, 471
wraptable, 471

X

xy, 291

Indice delle classi

A

arstexnica, 83
article, 93, 112, 465, 635, 638

B

beamer, 113, 323, 324
book, 19, 36, 93, 108, 111–114, 118–120, 127, 137, 140, 168, 246, 298, 300, 421, 446, 460, 466, 468–470, 479, 480, 636, 638, 647–649, 678, 679

L

layaureo, 298
letter, 112, 628, 635, 638
ltnews, 112
ltxdoc, 112, 638
ltxguide, 112

M

memoir, 115, 116, 154, 246, 248, 299, 388, 439, 449, 460, 461, 464, 466, 468, 469, 477
minimal, 112, 565, 638

N

ncc, 116

O

octavo, 36, 113, 298, 451, 454
ottavo, 298

P

proc, 112, 638

R

report, 93, 112, 118, 638

S

scrartcl, 114
scrbook, 114
scrlttr2, 114, 115
scrreprt, 114
slides, 113, 321, 322, 638
suftesi, 36

T

toptesi, 118

Indice dei file

Estensioni

,map, 381
,tfm, 377
.afm, 373, 383
.aux, 103, 685, 687
.bat, 88, 495
.bbl, 234, 685
.bib, 240
.bmp, 218
.bst, 232, 239
.dat, 572
.def, 587
.doc, 500–503, 506–510, 512, 598
.docx, 512
.dtx, 415, 435
.dvi, 245, 394, 395, 639, 685
.engine, 215
.eps, 210, 216, 218, 219, 222, 226
.fd, 378, 387, 388
.fmt, 79
.fodt, 516
.gif, 612
.glo, 246, 685
.gls, 246
.gst, 248
.icc, 489
.icm, 489
.idx, 242, 246, 247, 685
.ind, 243, 246, 247, 685
.ini, 81
.ist, 247
.jpg, 218, 219, 224
.lof, 635, 685
.log, 76, 80, 90, 103, 228, 259, 390, 558, 559, 614, 617, 620–622, 680, 685, 688
.lot, 635, 685
.ltx, 80, 102, 685
.lyx, 510
.map, 378
.mf, 89, 396
.mp, 212
.mps, 90, 212
.odt, 512
.otf, 373
.pdf, 210, 212, 216, 218, 219, 222, 226, 245, 395, 501, 579, 598, 639
.pfa, 383
.pfb, 373, 381, 383, 384, 491
.png, 218, 219, 612
.prj, 68
.ps, 210, 218, 226, 395, 496, 639
.rtf, 501, 507, 512
.scriv, 517
.sty, 126, 331, 378, 405
.svg, 210, 217, 222
.sxw, 512
.tex, 46, 47, 56, 77, 80, 91, 102, 223, 494, 496, 507–510, 512, 578, 579, 581, 585, 587, 590–593, 598, 599, 612, 628, 682, 685, 706
.tfm, 373, 382, 383, 396
.toc, 635, 686

.ttf, 373, 384
.txt, 500, 501
.vf, 377, 383
.vpl, 383
.xml, 512
.xmpdata, 494
.zip, 76

A

auctex, 45
Autorun.inf, 42

B

beamerexample2.article.pdf, 324
beamerexample2.beamer.pdf, 324
beamerexample5.pdf, 324
beameruserguide.pdf, 324
Bibliografia.bib, 239
bk10.clo, 446, 456, 648
bohahyph.tex, 571
book.cls, 127, 415

C

classic.ist, 247
clsguide, 113
cmr10.300pk, 382

D

dvipdfm.map, 374, 384
dvipsnam.def, 703

E

ECI-RGB.V1.0.icc, 489, 494
encguide.pdf, 585
endnote.tex, 145
endnotes.sty, 145
esempio1.dvi, 97
esempio1.tex, 97
euler.pdf, 365

F

fntguide.dvi, 704
fntguide.pdf, 364
fontmath.ltx, 365
fonts/map/, 386
fourier-doc-en.pdf, 365

G

gfsartemisia, 310
gfsbaskerville, 310
gfsbodoni, 310
gfscomplutum, 310
gfsdidot, 310
gfsporson, 310
gfssolomos, 311
gglo.ist, 248
gloss-greek.ldf, 571
grfguide.pdf, 701
GuidaGuIT.tex, 494
GuidaGuIT.xmpdata, 494

H

hyph-it.tex, 564
hyph-quote-it.tex, 564, 565

I

ind.dvi, 242
it_IT.aff, 67
it_IT.dic, 67
italian.ldf, 570, 617
ithyph.tex, 564

L

language-local.dat, 572
language-local.def, 572
latex.fmt, 80, 84, 245, 337
latex.ltx, 79, 81, 399, 649
latin.ldf, 570
latin1.def, 585, 586
libretto.pdf, 442

loadhyph-it.tex, 564
local.map, 386, 397

M

MacroGuida.sty, 36, 187, 402, 410, 418, 427
makebst.tex, 233, 237
makeindex.dvi, 242
mathdesign-doc.pdf, 365
Mathmode.pdf, 295
memdesign, 116
memman.gst, 248
merlin.mbs, 233
mf, 382
microtype.cfg, 339
miofont.mf, 382
missfont.log, 90
mt-MinionPro.cfg, 339
mymacros.sty, 127, 421, 424

N

neohellenic, 310
nuovateoria.tex, 104

O

ot1enc.def, 589

P

pag.fd, 318
pagk8r.pfb, 318
PDA_def.ps, 496
pdf2pdfa, 496
pdf2pdfa.bat, 495
pdflatex.fmt, 245
pdftex.map, 318, 374, 377, 384
pdfx.sty, 494
pict2e.cfg, 178
plain.bst, 232
plain.tex, 79, 81, 82, 399
protext-setup-en.pdf, 42
ps2pdfa, 496
ps2pdfa.bat, 496
ps4pdf.engine, 215
psfonts.map, 374, 384
psnfss2e.pdf, 375

S

scrguien.pdf, 464
segnature.pdf, 442
srcguide.gst, 248

T

t1enc.def, 586, 588
t1lmr.fd, 388
tds.pdf, 48, 52
tex, 517, 518, 520, 522, 524
texlive2013-20130530.iso, 44
TeXworks.app, 76
tiger.eps, 216
tttomia.cls, 127
txfonts, 311
txfontsx, 311
txt, 515–517

U

uni3.map, 381
uni3.pfb, 381
uni3.tfm, 381
unsrt.bst, 232
updmap.cfg, 385, 386, 396, 490
updmap.map, 397

Indice dei pacchetti

A

afterpage, 167, 176, 427, 428
alltt, 640, 650
amscd, 290
amsfont, 641
amsfonts, 254, 271, 272, 364, 640
amsmath, 117, 173, 174, 251, 254, 257, 259, 261, 263–265, 269, 271, 273, 277, 280–282, 284, 286–288, 290, 295, 363, 365, 370, 483, 484, 631, 640, 652–654, 704
amssymb, 271, 640, 641
amsymb, 363
array, 157, 159, 168–170, 173, 431, 683
auctex, 65, 70, 75
auto-pst-pdf, 185
avant, 318

B

babel, 26, 27, 94, 99, 102, 116, 122, 132, 133, 149, 265–267, 290, 291, 299–301, 307, 315, 328, 334, 367, 369, 400, 404, 410, 415, 417, 422, 505, 558, 562, 569, 571, 617, 635, 640, 651–653
beamer, 113, 323–325, 329, 331, 332
bgreek, 300
biblatex, 232, 239, 240
bm, 286, 363
bohairic, 371
booklet, 442, 443
booktabs, 161, 164

C

calc, 658
caption, 114, 154, 155, 464
circuitikz, 186
ClassicThesis, 36, 115, 117, 479
color, 322, 464, 475, 490, 640, 702
combelow, 589
cool, 483
coptic, 313, 371
crop, 443
csquotes, 239
curve2e, 328, 331

D

dcolumn, 168

E

ecclesiastic, 371
edmac, 312
edstanza, 312
eledmac, 312
endnotes, 145, 312
epsconversion, 216

epstopdf, 216
etex, 694
etoolbox, 318, 423, 560, 617, 667, 670–672
euler, 365
exercise, 149

F

fancyhdr, 114, 446, 447, 459, 464, 642
fixltx2e, 468, 675
float, 176, 177, 470, 472
floatrow, 177
fontenc, 328, 557, 579, 581, 706
fontspec, 86, 318
fourier, 365, 376, 410, 459
frontespizio, 117, 118

G

geometry, 114, 460, 461, 464, 643
ghostscript, 54
glossaries, 244, 245
glossary, 245
graphics, 475, 641, 702
graphicx, 166, 209, 212, 221, 330, 403, 431, 475, 490, 641, 702
guit, 94, 328

H

hyperref, 240, 406, 493, 494, 510
hyphenator.js, 557

I

IEEEtrantools, 173
ifthen, 471, 641, 667–669, 671
ImageMagik, 220
imakeidx, 248, 249, 641, 688, 689
imakeidxx, 247
indentfirst, 458
indextools, 247–249, 641, 688, 689
inputenc, 59, 60, 75, 328, 505, 557, 579, 581, 591, 593, 598, 634, 682, 706

K

koma-script, 114
kpfonts, 305, 376
kuvio, 290

L

latexsym, 254, 641
layaureo, 119, 120, 452, 453, 480
layout, 122
ledmac, 312
lettrine, 402, 403, 459
longtable, 167, 168
ltxsymb, 271

M

MacTeX.pkg, 46
makeidx, 243, 641, 688
mathdesign, 365, 367, 375
mathenv, 173
mdwtab, 173
metre, 313
mfnfss, 306
microtype, 333, 334, 338–341, 565, 568, 569, 602
morefloats, 676, 680
multicol, 421, 643, 675
multirow, 157

N

natbib, 232, 233, 240
newpx, 375
newpxmath, 360, 361, 365
newpxtext, 360, 365
newtx, 375
newtxmath, 360, 361, 365, 375
newtxtext, 360, 365, 375

O

opcit, 312

P

pdfcomment, 214
pdfpages, 215, 442, 443
pdfscreen, 323
pdfslide, 323
pdfx, 491, 493, 494
pgf, 185, 186, 188, 209, 323, 324, 328, 701
pgfplots, 91
pict2e, 122, 178, 182, 183, 185, 326, 328, 331, 641, 701
Pifonts, 375
poemscol, 312
polyglossia, 27, 315, 571
powerdot, 323, 332
ppower4, 323
prosper, 323, 332
psfrag, 189
pst-pdf, 185
PSTricks, 185, 214, 217, 332, 701
PSTriks, 215
pxfonts, 359, 361, 365, 375

R

rotating, 167
rotfloat, 177

S

sapthesis, 117
sectsty, 464
seminar, 323
showidx, 641
SIstyle, 267
SIunits, 267
siunitx, 164, 267, 268
suftesi, 117, 299
supertabular, 167, 168
svg, 217, 222

T

tabmac, 312
tabu, 164, 168, 174, 288, 289
tabularx, 162, 164, 168
technica, 299–301
TesiClassica, 117
TesiModerna, 117
testhyphens, 558
teubner, 301, 302, 308, 309, 311–313, 370
texpower, 323
textcomp, 367, 510, 580, 581, 586, 587
TikZ, 91
tikz, 187, 290
titlesec, 464, 465
tocloft, 407, 408, 410, 464
topcapt, 168
TOPtesi, 117, 118
trace, 619, 620, 624, 625
txfonts, 359, 361, 365, 375
type1ec, 638
typearea, 114, 115, 454, 460, 464

U

unicode-math, 86
units, 267
utopia, 365

V

varioref, 122, 141

W

widetable, 164
wrapfig, 471, 472

X

xcolor, 164, 323, 324, 490, 702
xypic, 133, 290, 291

Indice dei programmi e delle distribuzioni

Simboli

TEXShop, 73

A

AbiWord, 507, 508, 510, 512
Acrobat Professional, 488, 493, 498
Adobe Acrobat, 55, 221
Adobe Reader, 55, 69, 221, 490, 501
afm2tfm, 383
Aleph, 86, 87
aleph, 91
Anteprima, 55, 221, 511
Aquamacs, 75, 76, 585, 591, 592, 598
Aquamcas, 592
arara, 90
arlatex, 91
asy, 91
Asymptote, 91
auctex, 71, 76
autoinst, 383, 384

B

bash, 397
BibDesk, 46, 89, 233
biber, 89, 141, 231, 232, 240, 244, 687
BIBTEX, 72, 89, 141, 231, 232, 234–240, 244, 312, 406, 685–687
bibtex8, 89
bibtexu, 89

C

Character map, 57
charco, 597
context, 91, 513, 601
context-mk-iv, 590
convert, 220
CygWin, 69, 382, 507, 597

D

dbcontext, 513
dblatex, 512, 513
Draw, 511
dvipdfm, 78, 89, 214, 215, 332, 384, 487, 490
dvipdfmx, 89, 214, 215, 374, 381
dvips, 54, 55, 78, 89, 178, 185, 189, 214, 215, 332, 374, 381, 384, 395, 490, 513, 703

E

ebb, 214
elatex, 82

emacs, 45, 65, 68, 70, 71, 75, 76, 512, 585, 592
eps2jpg, 218
eps2pdf, 218
eps2png, 218
epspdf, 218
epstopdf, 89, 218
ε-TEX, 409
etex, 80, 82, 86, 91, 245, 361, 431, 455, 658, 663
EurKey, 58
Excel, 200
excel, 511

F

fmtutil, 83, 90
fmtutl-sys, 90
FontForge, 86, 377, 381, 383
fontinst, 383

G

gedit, 72
gedit-LaTeX-plugin, 72
ghostscript, 42, 54, 218, 487, 488, 493, 495, 496
ghostview, 42, 54, 218
gimp, 55, 219, 220
gnome, 72
gnuplot, 195, 198, 200
GraphicConverter, 220
gs, 214, 216
gv, 54
gview, 54, 55, 218, 220, 225, 226, 228

I

iconv, 594, 597
ImageMagik, 220
InDesign, 4
inkscape, 217, 490

J

JabRef, 233
Jabref, 89
java, 397
jpegtops, 55, 219

K

Kile, 45, 69–73, 76
kpathsea, 86
kpseaccess, 91
kpsepath, 91
kpsereadlink, 91
kpsestat, 91
kpsetool, 91
kpsewhere, 91
kpsewhich, 48, 91, 385
kpsexpand, 91
Kword, 507, 508, 510, 512

L

lamed, 91
latex, 46, 65, 78–81, 91, 214–216, 394, 405, 498, 509, 510, 513
LaTeXiT, 46, 47
latexmk, 90
LED, 68, 69, 84
Libre Office, 518
LibreOffice, 577
lua, 86, 87
lualatex, 87, 88, 91, 503
luatex, 85–87, 91, 503, 590, 601
LyX, 77, 498, 509, 510, 512

M

MacTEX, 45, 88, 233
makeindex, 89, 242–249, 401, 685, 688, 689
Mappa dei caratteri, 57
Marked2, 517
METAPOST, 86, 189, 210, 212, 214, 216, 219

mftrace, 382, 383, 396
MiKTEX, 42, 43, 48, 49, 51–53, 66–70, 88, 89, 97, 122, 247, 377–380, 386, 387, 392, 394, 569, 570, 613, 667
MiKTeX Maintainance, 53
MiKTeX Settings, 53
mktexlsr, 91
mllatex, 91
mltex, 91
mptopdf, 89, 219
MultimarkdownComposer, 517

N

NeoOffice, 557
Notepad, 69, 396

O

Omega, 86
Open Office, 518
Open XML converter, 512
OpenOffice, 557
OpenOffice.org, 511

P

Pages, 518
Paint, 219
Pandoc, 499, 515
patgen, 572, 573
pdfcrop, 90, 221
pdfelatex, 82
pdfetex, 80, 82
pdfimport, 511
pdfLaTeX, 329
pdflatex, 46, 66, 71, 75, 78–81, 86–88, 91, 187, 215–218, 222, 305, 309, 311, 313, 318, 319, 334, 353, 371, 394, 395, 405, 442, 488, 489, 493, 495, 498, 500, 505, 506, 508–510, 513, 558, 560, 568, 572, 578, 579, 586, 587, 590, 598, 600, 612, 641, 685, 688
pdftex, 79, 80, 82, 84–87, 91, 128, 180, 189, 215, 216, 245–247, 334, 337, 361, 374, 384, 399, 431, 443, 455, 456, 471, 487, 490, 569, 601–603, 605–607, 658, 663–666, 692
pdftex.exe, 247
pdftops, 219
perl, 397
potrace, 382
Preview, 55, 221, 511
ProTEXt, 42, 68
ps2pdf, 55, 78, 215, 218, 332, 495
ps4pdf, 89, 215, 217
pstopdf, 89
ptex, 91
Python, 72
python, 382, 397

Q

QuarkXPress, 4

R

rtf2latex2e, 507

S

Saxon, 512
Scribus, 3, 4
Scrivener, 517–520, 522, 524
Skim, 55, 75, 76, 511
StarOffice, 511
SumatraPDF, 42, 66

T

Task Manager, 614
tex, 79, 80, 82, 84, 86, 91, 215, 246, 399, 455, 456, 570, 601

TeX Live Manager, 53
TeX Live Utility, 46
TeX4ht, 499
texdoc, 48, 91, 476
TEX Live, 41–46, 48–54, 66, 68, 70, 71, 216–218, 244, 247, 248, 295, 299, 306, 313, 360, 368, 377–380, 384–387, 392, 397, 477, 569, 571, 572
TeXmacs, 77
TeXmaker, 45, 66, 600
Texmaker, 66, 68, 70, 72, 73, 77, 585
TeXnicCenter, 68–70
Texpad, 524
TeXShop, 66, 67, 72, 74, 109, 524
TEXShop, 46, 64, 72, 73, 75–77, 84, 97, 215, 219, 314, 315, 585, 588, 591, 592, 598, 599, 613, 614, 616
TeXStudio, 45
TeXstudio, 42, 65, 66, 68–70, 72, 77, 109, 233, 585, 592, 600
TextEdit, 512, 518
TextPad, 69
TeXtures, 77
TextWrangler, 76
TeXWorks, 524
TeXworks, 45, 46, 66, 67, 72, 75–77, 315, 585, 588, 591–594, 597–600, 613
tlmgr, 44, 45, 53
ttf2tfm, 384

U

uninstl.bat, 43
updmap, 90, 379, 384, 490
updmap-sys, 90, 384, 385

V

vim, 70, 396
vptovf, 383

W

Win-iconv, 597
WinCDEmu, 43
WinEdt, 68, 69, 219, 503
Word, 518
Word2LyXMacro, 509
word2tex, 507
Writer, 509, 511, 518, 577
Writer2LaTeX, 509, 511
Writer4LaTeX, 509

X

xdvi, 73, 97, 374, 394
xdvipdfmx, 89
xelatex, 27, 86–88, 91, 309, 310, 314–316, 318, 319, 503, 558
Xemacs, 71
xetex, 86, 91, 503, 590, 601
xfig, 198
xindy, 89
xpdf, 86

Y

YAP, 97, 374, 394